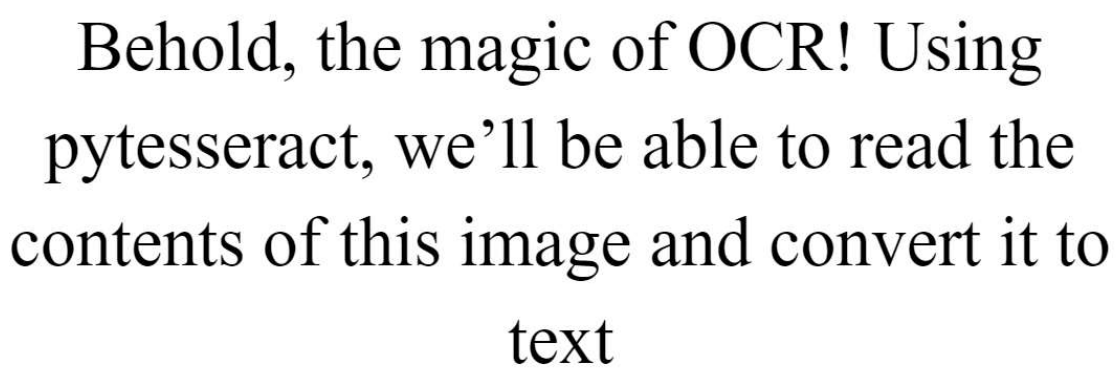
In [1]:
```
from PIL import Image
image=Image.open('readonly/text.png')
display(image)
```



Behold, the magic of OCR! Using pytesseract, we’ll be able to read the contents of this image and convert it to text

In [2]:
```
import pytesseract
dir(pytesseract)
```

Out[2]:
```
['Output',
 'TesseractError',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__path__',
 '__spec__',
 'get_tesseract_version',
 'image_to_boxes',
 'image_to_data',
 'image_to_osd',
 'image_to_pdf_or_hocr',
 'image_to_string',
 'pytesseract']
```

In [3]:
```
help(pytesseract.image_to_string)
```

```
Help on function image_to_string in module pytesseract.pytesseract:

image_to_string(image, lang=None, config='', nice=0, output_type='string')
    Returns the result of a Tesseract OCR run on the provided image to string
```

In [4]:
```
help(Image.Image.resize)
```

```
Help on function resize in module PIL.Image:

resize(self, size, resample=0, box=None)
    Returns a resized copy of this image.

    :param size: The requested size in pixels, as a 2-tuple:
       (width, height).
    :param resample: An optional resampling filter.  This can be
       one of :py:attr:`PIL.Image.NEAREST`, :py:attr:`PIL.Image.BOX`,
       :py:attr:`PIL.Image.BILINEAR`, :py:attr:`PIL.Image.HAMMING`,
       :py:attr:`PIL.Image.BICUBIC` or :py:attr:`PIL.Image.LANCZOS`.
       If omitted, or if the image has mode "1" or "P", it is
       set :py:attr:`PIL.Image.NEAREST`.
       See: :ref:`concept-filters`.
    :param box: An optional 4-tuple of floats giving the region
       of the source image which should be scaled.
       The values should be within (0, 0, width, height) rectangle.
       If omitted or None, the entire source is used.
    :returns: An :py:class:`~PIL.Image.Image` object.
```

In [6]:
```
import inspect
src=inspect.getsource(pytesseract.image_to_string)
print(src)
```

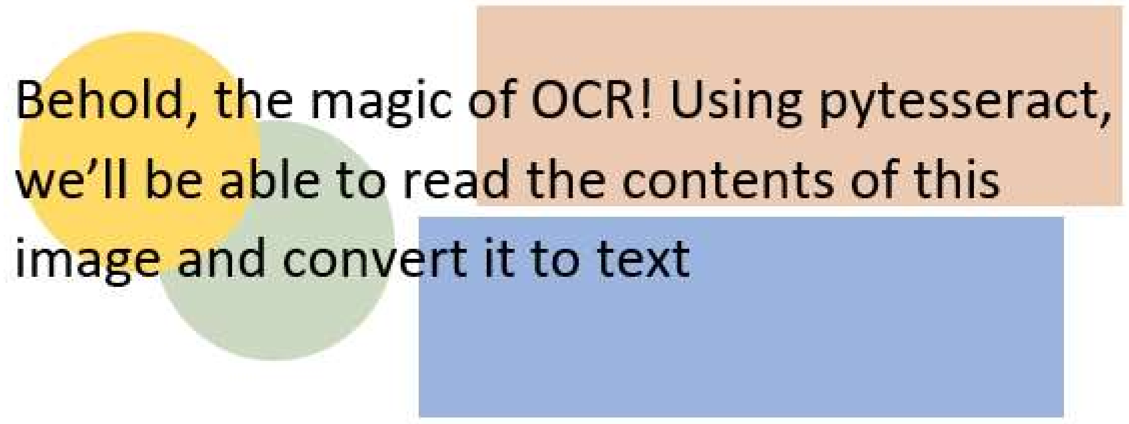
```
def image_to_string(image,
                    lang=None,
                    config='',
                    nice=0,
                    output_type=Output.STRING):
    '''
    Returns the result of a Tesseract OCR run on the provided image to string
    '''
    args = [image, 'txt', lang, config, nice]

    return {
        Output.BYTES: lambda: run_and_get_output(*(args + [True])),
        Output.DICT: lambda: {'text': run_and_get_output(*args)},
        Output.STRING: lambda: run_and_get_output(*args),
    }[output_type]()
```

In [7]:
```
#append 2 ? at the end of a function
pytesseract.image_to_string??
```

In [9]:
```
text=pytesseract.image_to_string(image)
print(text)
```

```
Behold, the magic of OCR! Using
pytesseract, we’ll be able to read the
contents of this image and convert it to
text
```

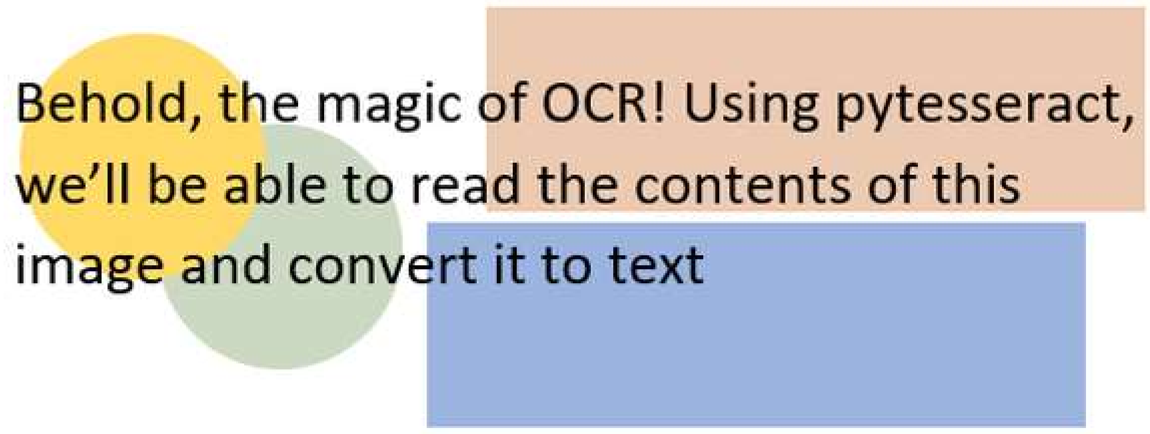
In [10]:
```
img=Image.open('readonly/Noisy_OCR.PNG')
display(img)
```


Behold, the magic of OCR! Using pytesseract, we’ll be able to read the contents of this image and convert it to text

In [11]:
```
text=pytesseract.image_to_string(Image.open('readonly/Noisy_OCR.PNG'))
print(text)
```

```
e magic of OCR! Using pytesseract,
le to read the contents of this



d convert it to text
```

In [12]:
```
import PIL
basewidth=600
img=Image.open('readonly/Noisy_OCR.PNG')
wpercent=(basewidth/float(img.size[0]))
hsize=int(float(img.size[1]*float(wpercent)))
img=img.resize((basewidth,hsize), PIL.Image.ANTIALIAS)
img.save('resized_noise.png')
display(img)
text=pytesseract.image_to_string(img)
print(text)
```



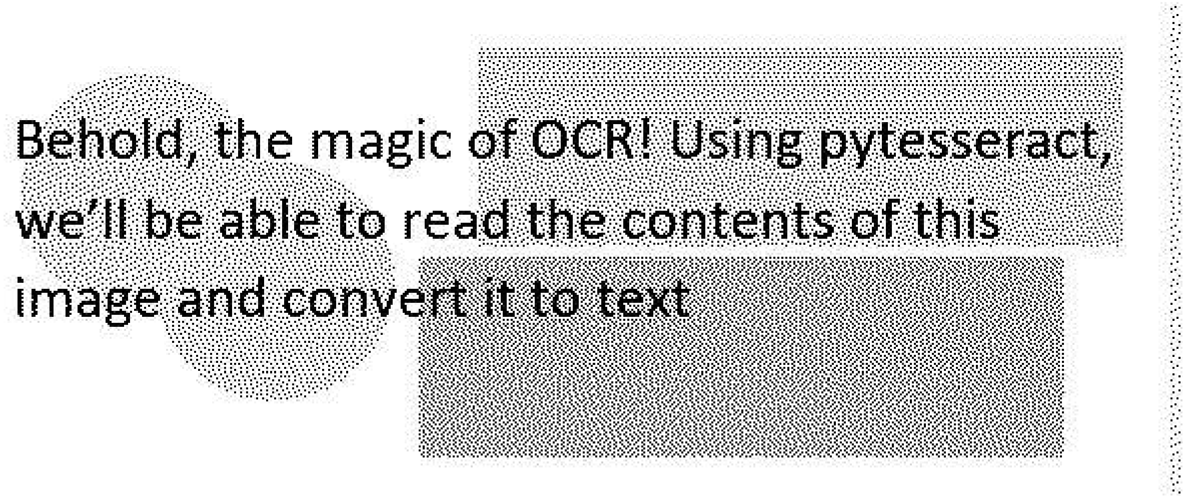
Behold, the magic of OCR! Using pytesseract, we'll be able to read the contents of this image and convert it to text

```
e magic of OCR! Using pytesseract,
le to read the contents of this
d convert it to text
```

In [13]:
```
img=Image.open('readonly/Noisy_OCR.PNG')
img=img.convert('L')
img.save('greyscale_noise.png')
text=pytesseract.image_to_string(img)
print(text)
```

```
Behold, the magic of OCR! Using pytesseract,
we'll be able to read the contents of this
image and convert it to text
```

In [16]:
```
#Binarize
img=Image.open('readonly/Noisy_OCR.PNG').convert('1')
image.save('black_and_white_noisy.png')
display(img)
```

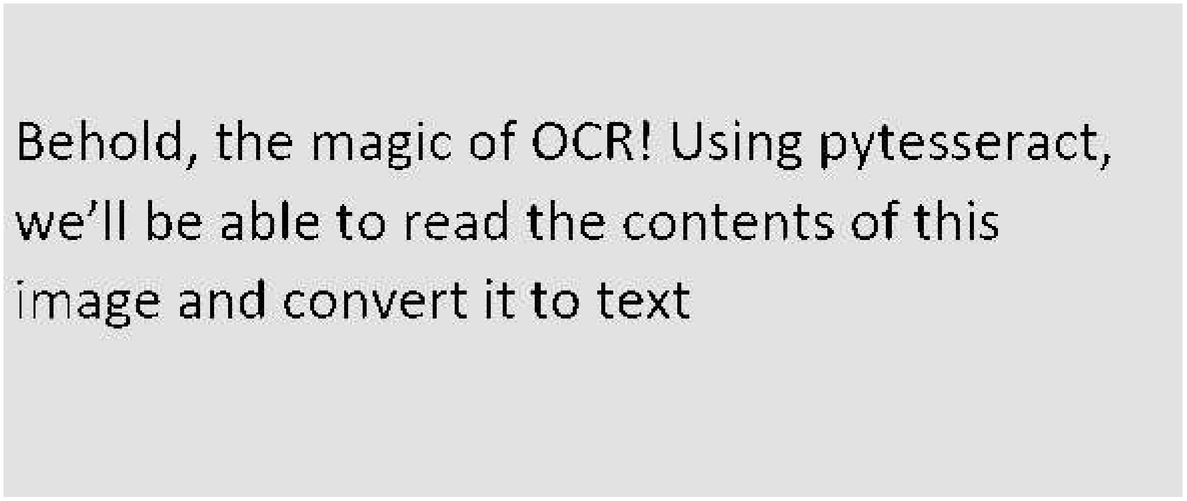

Behold, the magic of OCR! Using pytesseract, we'll be able to read the contents of this image and convert it to text

In [21]:
```
def binarize(image_to_transform, threshold):
    output_image=image_to_transform.convert('L')
    for x in range(output_image.width):
        for y in range(output_image.height):
            if output_image.getpixel((x,y))<threshold:
                output_image.putpixel((x,y),0)
            else:
                output_image.putpixel((x,y),225)
    return output_image

for thresh in (0,257,64):
    print("Trying with threshold"+str(thresh))
    display(binarize(Image.open('readonly/Noisy_OCR.PNG'), thresh))
    print(pytesseract.image_to_string(binarize(Image.open('readonly/Noisy_OCR.PN
```

```
Trying with threshold0
```

```
Trying with threshold257
```

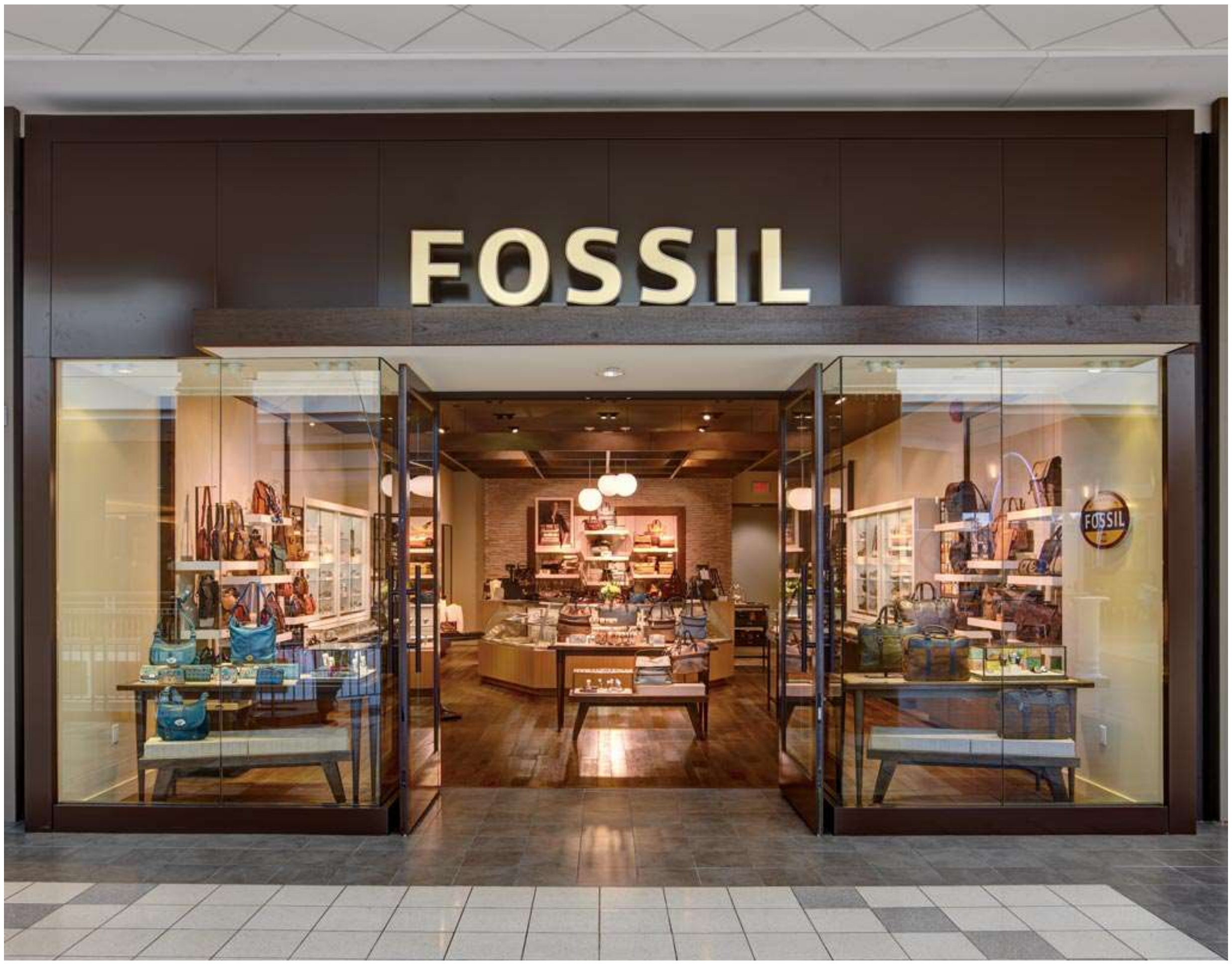
```
Trying with threshold64
```



Behold, the magic of OCR! Using pytesseract,
we'll be able to read the contents of this
image and convert it to text

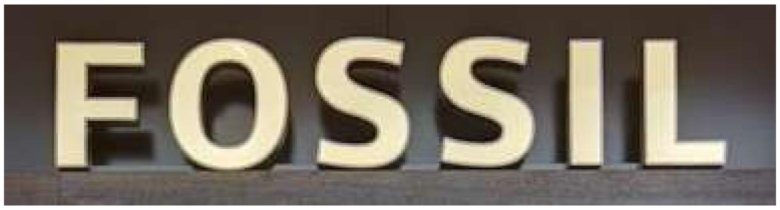
```
Behold, the magic of OCR! Using pytesseract,
we'll be able to read the contents of this
image and convert it to text
```

In [2]:
```python
from PIL import Image
import pytesseract

image=Image.open('readonly/storefront.jpg')
display(image)
pytesseract.image_to_string(image)
```



Out[2]:
```
''
```

In [3]:
```
bounding_box=[315,170,700,270]
title_image=image.crop(bounding_box)
display(title_image)
pytesseract.image_to_string(title_image)
```



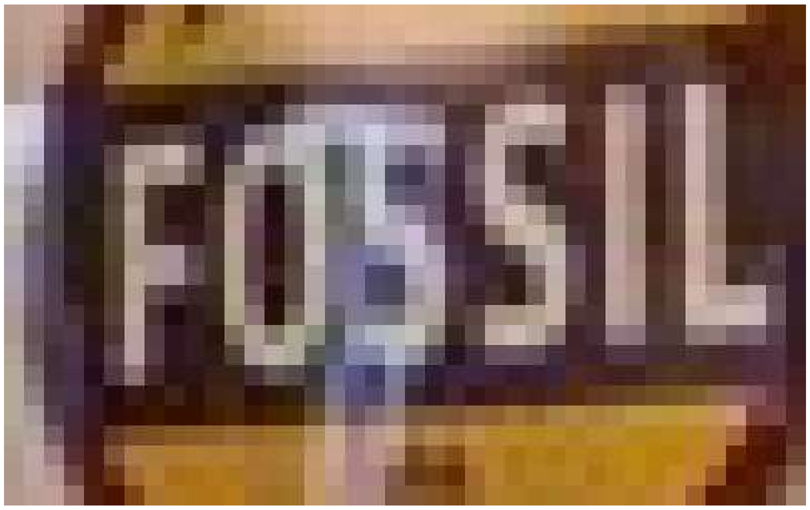
Out[3]:
```
'FOSSIL'
```

In [4]:
```
bounding_box=[900,420,940,445]
little_sign=image.crop(bounding_box)
display(little_sign)
```

In [5]:
```
new_size=(little_sign.width*10,little_sign.height*10)
help(little_sign.resize)
```

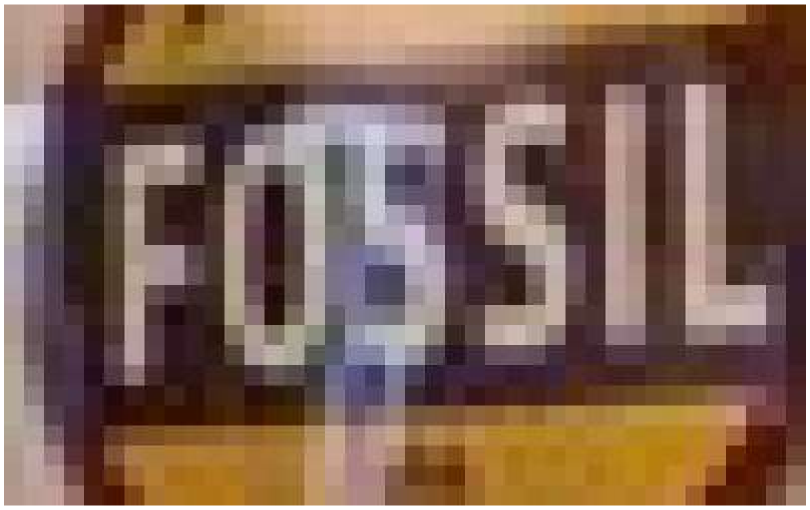
```
Help on method resize in module PIL.Image:

resize(size, resample=0, box=None) method of PIL.Image.Image instance
    Returns a resized copy of this image.

    :param size: The requested size in pixels, as a 2-tuple:
       (width, height).
    :param resample: An optional resampling filter.  This can be
       one of :py:attr:`PIL.Image.NEAREST`, :py:attr:`PIL.Image.BOX`,
       :py:attr:`PIL.Image.BILINEAR`, :py:attr:`PIL.Image.HAMMING`,
       :py:attr:`PIL.Image.BICUBIC` or :py:attr:`PIL.Image.LANCZOS`.
       If omitted, or if the image has mode "1" or "P", it is
       set :py:attr:`PIL.Image.NEAREST`.
       See: :ref:`concept-filters`.
    :param box: An optional 4-tuple of floats giving the region
       of the source image which should be scaled.
       The values should be within (0, 0, width, height) rectangle.
       If omitted or None, the entire source is used.
    :returns: An :py:class:`~PIL.Image.Image` object.
```

In [6]:
```
display(little_sign.resize(new_size,Image.NEAREST))
```

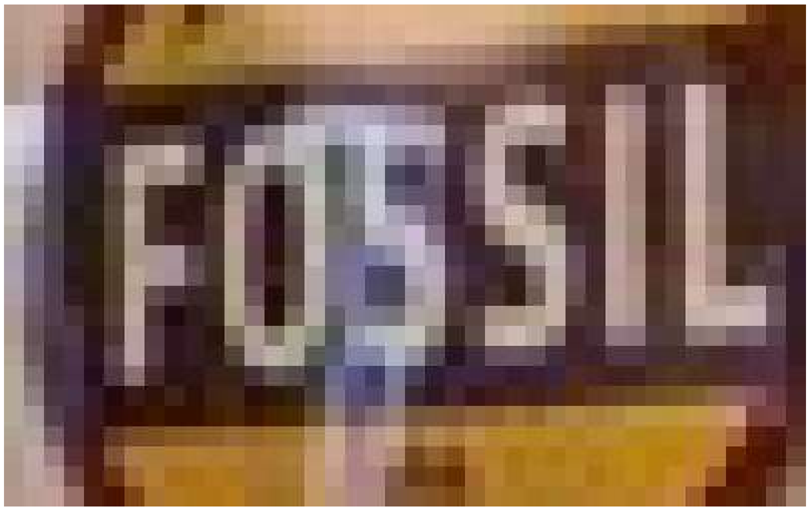


In [7]:
```
options=[Image.NEAREST, Image.BOX, Image.BILINEAR, Image.HAMMING, Image.BICUBIC,
for option in options:
    print(option)
    display(little_sign.resize(new_size,option))
```

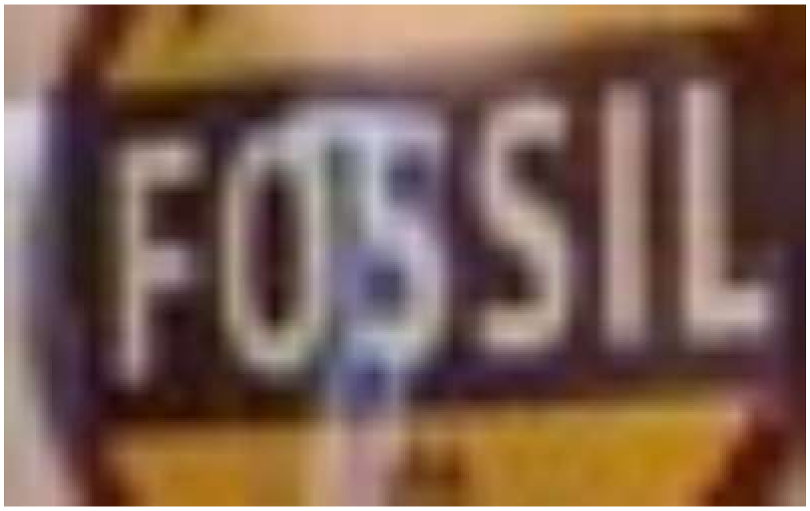
```
0
```



```
4
```



```
2
```



5

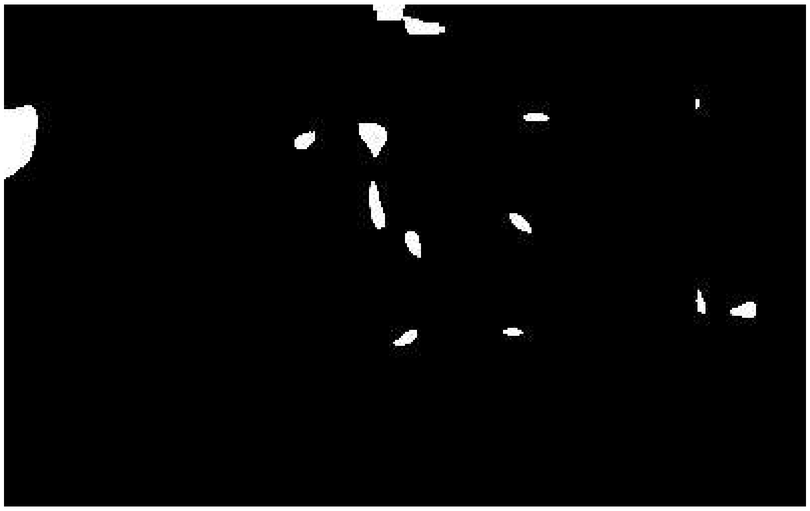
3

1

In [8]:
```
bigger_sign=little_sign.resize(new_size, Image.BICUBIC)
pytesseract.image_to_string(bigger_sign)
```

Out[8]:
```
''
```

In [9]:
```
def binarize(image_to_transform,threshold):
    output_image=image_to_transform.convert('L')
    for x in range(output_image.width):
        for y in range(output_image.height):
            if output_image.getpixel((x,y))<threshold:
                output_image.putpixel((x,y),0)
            else:
                output_image.putpixel((x,y),255)
    return output_image
binarized_bigger_sign=binarize(bigger_sign,190)
display(binarized_bigger_sign)
pytesseract.image_to_string(binarized_bigger_sign)
```



Out[9]:
```
'Lae'
```

In [15]:
```
eng_dict=[]
with open('readonly/words_alpha.txt','r') as f:
    data=f.read()
    eng_dict=data.split('\n')
for i in range(150,170):
    strng=pytesseract.image_to_string(binarize(bigger_sign,i))
    strng=strng.lower()
    import string
    comparision=''
    for character in strng:
        if character in string.ascii_lowercase:
            comparision=comparision+character

        if comparision in eng_dict:
            print(comparision)
```

```
f
fo
foss
fossil
s
si
f
fo
foss
fossil
f
fo
foss
fossil
g
ga
gas
g
ga
gas
s
sl
s
sl
s
si
sil
```

In [ ]:
```
#Jupiter widgets
from PIL import Image, ImageDraw
from ipywidgets import interact
image=Image.open('readonly/storefront.jpg')
@interact(left=100, top=100, right=200, bottom=200)

def draw_border(left,top,right,bottom)
```