Regular Expressions Part 2

Along with some other bits and pieces related to strings in Python

f-strings

f-strings are formatted strings in Python

We can insert other variables into a string, by prefacing the quoted string with an f, and then using curly brackets to indicate where the variables should be inserted.

In [1]:
```
a=3
b=47.5
stemp=f"While b moved up to {b}, a remains stubbornly fixed at {a}"
print(stemp)


# I use the variable name stemp constantly to mean string temporary,
```
```
While b moved up to 47.5, a remains stubbornly fixed at 3
```

Question/Action

Create an f string that inserts a persons first and last name and telephone number into a sentence

Create variables to hold the first and last name and the phone number

In [3]:
```
first_name = "Evelyn"
last_name = "Barragan"
telephone = "123-456-7891"

print(f"The person name is {first_name} {last_name} and their phone number i
```
```
The person name is Evelyn Barragan and their phone number is 123-456-7891
```

r-strings

An r-string in Python is a *raw* string in which backslashes are not treated escape characters, so that r"\d" is equivalent to "\r"

This can be helpful in creating regression strings

Compiling Regex patterns

In the re package, there is a function re.compile() that will take in a string specifying a regex expression and compile it into a *regex pattern object*

This is useful if you will be repeatedly using the same search pattern. It will cause the code to run faster.

It turns out the re.match and re.search will compile the pattern automatically, so just how much improvement one gets from this is perhaps limited.

In [3]:
```
import re
```

In [5]:
```
target=re.compile(r"\d{2,4}")

example_str="Joe picked 113 pears this morning"

print(re.search(target,example_str))
```

```
<re.Match object; span=(11, 14), match='113'>
```

In [6]:
```
# accessing the extracted value

re.search(target,example_str).group()
```

Out[6]:
```
'113'
```

Sidenote on AI

I used a version of the Gemmi AI to look up how to use the function re.extract()

Here is the response:

Okay, let's delve into the re.extract() function in Python's re module. It's a particularly interesting and sometimes tricky function, so I'll give you a thorough breakdown.

Purpose:

re.extract() attempts to find all occurrences of a pattern in a string and return them as a list. However, it differs significantly from other re functions like re.findall() or re.search(). It's designed to be more robust in certain situations, but it comes with some caveats.

Syntax:

re.extract(pattern, string, group=0, max repeats=0) Let's break down the parameters:

pattern: (Required) The regular expression pattern to search for. This is the same pattern you'd use in re.search(), re.findall(), etc.

string: (Required) The string to search within.

group: (Optional) This parameter controls how many capturing groups in the regular expression are returned.

0 (Default): Returns the entire match. 1: Returns the content of the first capturing group. 2: Returns the content of the second capturing group, and so on. If you specify a group number larger than the number of capturing groups in the pattern, you'll get a ValueError. max repeats: (Optional) Specifies the maximum number of times the pattern can be repeated. If set to a positive integer, re.extract() will attempt to find all occurrences up to that number of repeats. If not set, it attempts to find all occurrences.

Return Value:

If a match is found, re.extract() returns a list of the captured groups from the match. The order of the elements in the list corresponds to the order of the capturing groups in the regular expression. If no match is found, it returns None. Important Differences from re.findall():

re.findall() always returns a list of all matching strings, regardless of capturing groups. re.extract() only returns the captured groups, based on the group parameter. If you want the entire match, you must use group=0.

That all looks good, right? Drop it in the code in go.

But, it didn't work and when I looked up the re manual at

https://docs.python.org/3/library/re.html#functions

re.extract() does not exist- this is pure hallucination that looks real

and a couple of other LLMs confirmed that re.extract() does not exist.

I'm not making this stuff up, honest

References

See

https://docs.python.org/3/library/re.html#functions

and

Dave Child's regex cheatsheet- this explains a lot about how to create regex patterns- download a copy of this and hang on to it

https://cheatography.com/davechild/cheat-sheets/regular-expressions/

Question/Action

Here is a list of strings to work with

```
In [1]: str_list=["Hey, the cat ate my hamburger", "Hi, I'm in classroom 311 and got
                  "High, I'm in classroom 24 and got a 32 error on PowerPoint",
                  "This was much better than Cats, we will see it again and again","
                 "The dog would never steal my hamburger, he's too loyal"]

        str_list
```

```
Out[1]: ['Hey, the cat ate my hamburger',
         "Hi, I'm in classroom 311 and got a 404 error on Canvas",
         '867-5309',
         "High, I'm in classroom 24 and got a 32 error on PowerPoint",
         'This was much better than Cats, we will see it again and again',
         'My cell is 333-456-1212, call me',
         "The dog would never steal my hamburger, he's too loyal"]
```

Question

Use re and regex to find the string with the phone number of the form xxx-xxxx

```
In [13]: target=re.compile(r"\b\d{3}[-]\d{4}$")

         for s in str_list:
             if target.search(s):
                 print(s)
```

```
867-5309
```

Question

Use re and regex to find the string with the phone number of the form xxx-xxx-xxxx

```
In [14]: target=re.compile(r"\b\d{3}[-]\d{3}[-]\d{4}")

         for s in str_list:
             if target.search(s):
                 print(s)
```

```
My cell is 333-456-1212, call me
```

Question

Combine the previous questions into one regex expression that finds both phone numbers

Pull them out of the string and print them

```
In [19]: target=re.compile(r"(\b\d{3}-\d{4}\b)|(\b\d{3}-\d{3}-\d{4}\b)")

         for s in str_list:
             if target.findall(s):
                 print(re.search(target,s).group())
```

```
867-5309
333-456-1212
```

Question

find the sentence that ends with hamburger

```
In [43]: target=re.compile(r"hamburger$")

         for s in str_list:
             if target.search(s):
                 print(s)
```

```
Hey, the cat ate my hamburger
```

Question

Find all the strings that have the one cat in them with one search pattern

```
In [20]: target=re.compile(r"[Cc]ats?")

         for s in str_list:
             if target.findall(s):
                 print(s)
```

```
Hey, the cat ate my hamburger
This was much better than Cats, we will see it again and again
```

Question

Replace the word dog with wolverine

```
In [13]: target=re.compile(r"dog")

         for s in str_list:
             if target.findall(s):
                 print(re.sub(target, "wolverine", s))
```

```
The wolverine would never steal my hamburger, he's too loyal
```

Question

Find the values of the two error codes that were reported. Create one search pattern that finds both.

```
In [26]: # (?=error) means it must be followed by error but don't include error in re
         target=re.compile(r"\b(\d{2,3})\b(?= error)")

         for s in str_list:
             if target.findall(s):
                 print(re.search(target,s).group())
```

```
404
32
```

```
In [ ]:
```