Using Regular Expressions to Match Patterns

Using regular expression to match patterns is useful when you want to look for strings in a file with lots of information. For this expercise we will look at three scenarios and use regular expressions to match patterns. To do this we will first add the regular expressions package in python (re). Then for each instance we will simulate some data, build a regular expression to capture matches in the simulated data using the re.compile function, and test the regular expression with the filter function, which takes the variable name of the regular expression then a period then specification of how you want to match it, and then which list to look in.

Times

This pattern will match times after noon but before midnight when reported in 24-hour or "military" format.

First attempt

This first regeular expression (\d{1,2}\:\d{2}) captures times in general. It looks for 1 or 2 numerical characters, then a colon, and then 2 numerical characters. However, this does not solve the problem.

In [1]:
```
#add necessary packages
import re
#Simulate data
times=['00:15','01:15','02:15','3:15','4:15','05:15','06:15','07:15','08:15',
'09:15','10:15','11:15','12:15','13:15','14:15','15:15','16:15','17:15','18:1
5','19:15','20:15','21:15','22:15','23:15']
#Build expression
regex=re.compile('\d{1,2}\:\d{2}') #captures any time
#filter
print(filter(regex.match,times))
```

```
['00:15', '01:15', '02:15', '3:15', '4:15', '05:15', '06:15', '07:15', '08:1
5', '09:15', '10:15', '11:15', '12:15', '13:15', '14:15', '15:15', '16:15',
'17:15', '18:15', '19:15', '20:15', '21:15', '22:15', '23:15']
```

Second attempt

The next bit captures anything that starts with 1 or 2 and has 0-9 for the second digit. However, this still captures time before noon because it captures 10:xx and 11:xx.

In [2]:
```
#Build expression
regex2=re.compile('[1|2][0-9]:\d{2}')
#filter
print(filter(regex2.match,times))
```

```
['10:15', '11:15', '12:15', '13:15', '14:15', '15:15', '16:15', '17:15', '18:
15', '19:15', '20:15', '21:15', '22:15', '23:15']
```

Final Regular Expression

The combined used of the two regular expressions makes it so that the only digits you capture starting with a 1 are for 12:xx. Then, the print concetates the other times that you would want to capture.

In [3]:
```
#Build expression(s)
regex3=re.compile('[1][2-9]:\d{2}')
regex4=re.compile('[2][0-9]:\d{2}')
print(filter(regex3.match,times)+filter(regex4.match,times))
```

```
['12:15', '13:15', '14:15', '15:15', '16:15', '17:15', '18:15', '19:15', '20:
15', '21:15', '22:15', '23:15']
```

Genus/Species Names

This code will match genus species names expression in the format G. species (e.g. 'F. pennsylvanica'). This code looks for a capital letter [A-Z], then a period ., then a space \s, and then 2-25 lower case letters [a-z]{2-25}. Also, because we use .match in the filter function, it looks for instances where string starts with what we have specified.

In [4]:
```
#Simulate data
names=['Julio','C. elegens', 'F. pennsylvanica', 'J. E. Hoover', 'Horseraddis
h', 'H. sapien', 'G. maculatus', 'M. methanoregula', 'Rick', 'Eggs','J.i.b.b.
e.r.i.s.h']
#Build expression
regex=re.compile('[A-Z]\.\s[a-z]{2,25}')
#filter
print(filter(regex.match,names))
```

```
['C. elegens', 'F. pennsylvanica', 'H. sapien', 'G. maculatus', 'M. methanore
gula']
```

As you can see, it ignlored Julio because it needed a space. It ignored J. E. Hoover because the E. hoover wasn't at the beginning. Also it ignored all other instances that didn't match the regular expression.

Social Security Numbers

This code will match social security numbers in the proper US format (e.g 389-05-4771). This might be helpful if you had hacked into the aarp database and were trying to steal people's identities. One thing you would want is their social. This regular expression looks for three digits \d{3}, then a dash -, then two digits \d{2}, then a dash -, and lastly three more digits \d{3}, just like a social security number.

In [5]:
```
#simulate data
aarp=['389-05-4771','801-990-4372', 'Jeanne Romero-Severson', 'David Hyde', 'M
icky Mouse', '333-11-9909', '574-661-9080', '555-444-9999', '616-21-7703']
#Build expression
regex=re.compile('\d{3}\-\d{2}\-\d{4}')
#filter
print(filter(regex.match,aarp))
```

```
['389-05-4771', '333-11-9909', '616-21-7703']
```