03-Regular-Expressions

November 21, 2019

1 Search/replace in Regular expression

- `re.sub(regex, replacement, inputstring)` replaces all occurences of regex into replacement in inputstring. If no match, original string is returned

```
In [2]: import re
        print(re.sub(" +","|","onur:      tolga ,    sehit:ogl++"))
        print(re.sub("[:^.+;,/*-]","","onur: tolga, sehit++ogl/u"))
```

```
onur:|tolga|,|sehit:ogl++
onur tolga sehitoglu
```

- Replacement may refer to the matched groups as group id `\1, \2, \3`.
- `\g<1> \g<2>,...` can be used as well. `\g<0>` is the whole matched string.
- `\g<groupname>` syntax substitues the named groups.

```
In [10]: print(re.sub("^([A-Za-z]+) ([A-Za-z]+)$","\\2 \\1", "onur tolga"))
         print(re.sub("^([A-Za-z]+) ([A-Za-z]+)$","hello '\g<0>', how are you",
                      "onur tolga"))
         print(re.sub("^(?P<name>[A-Za-z]+) (?P<sname>[A-Za-z]+)$",
                      "\g<sname> \g<name> \g<name>\g<sname>", "onur tolga"))

         print(re.sub("^(?P<name>[a-z]+) (?P<mname>[a-z]+ )?(?P<sname>[a-z]+)$",
                     "\g<sname>, \g<name> (\g<mname>)",
                      "onur tolga sehitoglu"))
         print(re.sub("^(?P<name>[a-z]+) (?P<mname>[a-z]+ )?(?P<sname>[a-z]+)$",
                     "\g<sname>, \g<name> \g<mname>",
                      "onur sehitoglu"))
```

```
tolga onur
hello 'onur tolga', how are you
tolga onur onurtolga
sehitoglu, onur (tolga )
sehitoglu, onur
```

- Substitution can be a function. In that case, match object will be sent to the function as parameter. String returned by the function is the result of the substitution.

```
In [11]: # function substitution
         def convert(m):
             '''replace name middle surname as surname, n. m.'''
             print(m)
             res = m.group('sname')
             res += ', '
             res += m.group('name')[0] + "."
             if m.group('mname'):
                 res += " " + m.group('mname')[0] + "."
             return res

         print(re.sub("^(?P<name>[a-z]+) (?P<mname>[a-z]+ )?(?P<sname>[a-z]+)$",
                      convert, "onur tolga sehitoglu"))
```

```
<_sre.SRE_Match object; span=(0, 20), match='onur tolga sehitoglu'>
sehitoglu, o. t.
```

```
In [20]: ipmatch='([0-9]|[1-9][0-9]|1[0-9]{2,2}|2[0-4][0-9]|25[0-5])\\.([0-9]|[1-9][0-9]|1[0-9]{
         ipre = re.compile(ipmatch)

         def tolist(m):
             ret = ""
             for v in m.groups():
                 ret += str(int(v)+1) + "."
             return ret

         ipre.sub(tolist,"144.122.71.1")
```

```
Out[20]: '145.123.72.2.'
```

2 Input sanitization/validation

- Validate all user input
- validate as early as possible
- Validate in all interfaces, revalidate as necessary

```
In [7]: emailexp="^([a-zA-Z._0-9]+)@([a-z_0-9]+\.)+([a-z_0-9]+)$"
        emailvalidator=re.compile(emailexp)

        email=input()
        print(emailvalidator.match(email))
```

```
babayaro@gmail.com
<_sre.SRE_Match object; span=(0, 18), match='babayaro@gmail.com'>
```

```
In [3]: ipc="([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])"
        ipvalidator=re.compile('^(' + ipc + '\.){3}' + ipc +'$')
```

```
        ip=input()
        print(ipvalidator.match(ip))

144.122.104.11
<_sre.SRE_Match object; span=(0, 14), match='144.122.104.11'>
```