ΠΡΟΑΠΑΙΤΟΥΜΕΝΑ:

1. Python Advanced: Μάθημα 2 - Iterators

ΠΕΡΙΕΧΟΜΕΝΑ:

1. Flags και αντικείμενα match
2. Συναρτήσεις του re
    1. Ταιριάσματα: match(), fullmatch() και search()
    2. Χωρισμός: split() και findall()
    3. Αντικατάσταση: sub() και subn()

Κατερίνα Τ.

Σμαραγδένιος Χορηγός Μαθήματος

Θοδωρής Κ.

Σμαραγδένιος Χορηγός Μαθήματος

ΜΑΘΗΜΑ 8.2: Το module re (Β') — 1. Flags και αντικείμενα match

modules psounis

Οι **σημαίες (flags)** επηρεάζουν συναρτήσεις του re.

- (όπως η re.compile() που το δέχεται ως προαιρετικό 3ο όρισμα)

re	Ταίριασμα με:
A ή ASCII	Ταίριασμα μόνο για ASCII χαρακτήρες
I ή IGNORECASE	Το ταίριασμα γίνεται case-insensistive
L ή LOCALE	\w,\W,\b,\B εξαρτώνται από το LOCALE
M ή MULTILINE	^: ταιριάζει με αρχή κάθε γραμμής $: ταιριάζει με τέλος κάθε γραμμής
S ή DOTALL	Η τελεία ταιριάζει και με το χαρ/ρα αλλαγής γραμμής (default δεν ταιριάζει με το \n)
X ή VERBOSE	Επιτρέπει σχόλια σε κανονικές εκφράσεις

Παρατήρηση:

- Μπορούμε να θέσουμε πολλά flags ενώνοντας τα με | (λογικό or)

Παράδειγμα 1: flags

```
pattern = re.compile("A.*A", re.IGNORECASE)
res = matches("abba", pattern)
print(res)

res2 = re.findall(r"""
      ^\s*?   # start with some whitespace
      <(.+?)> # extract the first tag you find
      """, text, re.MULTILINE | re.VERBOSE)
```

Η κλάση match περιέχει πληροφορίες για το ταίριασμα που έγινε:

- Μέλη Αντικειμένου:

Χαρ/κό	Περιγραφή
re	Κανονική έκφραση που χρησιμοποιήθηκε
string	Κείμενο που έγινε το ταίριασμα της κ.ε.

- Μέθοδοι/Τελεστές για groups (εξαγωγή συμβολοσειρών):

Μέθοδος	Περιγραφή
group([g1, g2, …])	Επιστρέφει tuple με τα groups (g1, g2,… είναι είτε indexes, είτε λεκτικά που έχουμε καθορίσει)
groups()	Επιστρέφει tuple με όλα τα groups
groupdict()	Επιστρέφει λεξικό με τα groups (εφόσον έχουμε καθορίσει ονόματα στα groups)
[g1, g2, …]	Πρόσβαση ως πίνακας (g1, g2,… είναι είτε indexes, είτε λεκτικά που έχουμε καθορίσει)

- Μέθοδοι/Τελεστές για groups (εξαγωγή θέσεων ταιριάσματος):

Μέθοδος	Περιγραφή
start([group])	Θέση αρχής του ταιριάσματος του group
end([group])	Θέση τέλους του ταιριάσματος του group
span([group])	tuple: (start[group], end[group])

- και τα μέλη που σχετίζονται με groups:

lastindex, lastgroup	Αριθμός (ή αντ. όνομα) του τελευταίου group

Παράδειγμα 2: groups3.py (βλ. βίντεο)

ΜΑΘΗΜΑ 8.2: Το module re (Β') 2.1. Ταιριάσματα: match(), fullmatch() και search()

Συναρτήσεις που εντοπίζουν ταίριασμα της κ.ε. στη συμβ/ρα:

- **Ταίριασμα κ.ε. στην αρχή ή σε όλη τη συμβολοσειρά:**

Συνάρτηση/Μέθοδος	Επεξήγηση
match(pattern, text, flags)	Ελέγχει αν το pattern ταιριάζει στην αρχή της συμβολοσειράς (επιστρ. το τελευταίο ταίριασμα, αν υπάρχουν περισσότερα από ένα ταιριάσματα)
Pattern.match (text[, start[, end]])	Ομοίως, αλλά προαιρετικά, καθορίζουμε αρχή και τέλος μέρους του text, στο οποίο θα γίνει η αναζήτηση)
fullmatch(pattern, text, flags)	Ελέγχει αν το pattern ταιριάζει με όλη τη συμβολοσειρά
Pattern.fullmatch (text[, start[, end]])	Ελέγχει αν το pattern ταιριάζει με όλη τη συμβολοσειρά (προαιρετικά, καθορίζουμε αρχή και τέλος μέρους του text, στο οποίο θα γίνει η αναζήτηση)

- αν υπάρχει ταίριασμα, επιστρέφεται αντικείμενο match, αλλιώς επιστρέφεται None.

Παράδειγμα 3: match_fullmatch

```
print(re.match(r"Comp.", text))
print(re.match(r"Sci.", text))
print(re.compile(r"Sci.").match(text, 9))

print(re.fullmatch(r"Comp.*", text))
print(re.compile(r"Sci\w+").fullmatch(text, 9, 14))
```

Ταίριασμα οπουδήποτε στη συμβολοσειρά:

re	Ταίριασμα με:
search(pattern, text, flags)	Ελέγχει αν το pattern ταιριάζει οπουδήποτε στη συμβολοσειρά (επιστρ. το πρώτο ταίριασμα)
Pattern.search (text[, start[, end]])	Ελέγχει αν το pattern ταιριάζει οπουδήποτε στη συμβολοσειρά (προαιρετικά, καθορίζουμε αρχή και τέλος μέρους του text, στο οποίο θα γίνει η αναζήτηση) (επιστρ. το πρώτο ταίριασμα)

- αν υπάρχει ταίριασμα, επιστρέφεται αντικείμενο match, αλλιώς επιστρέφεται None.

Παράδειγμα 4: search.py

```
text = "Computer Science is no more about computers " + \
    "than astronomy is about telescope"
print(re.search(r"comp.", text))
print(re.search(r"Sci.", text))
print("="*20)
pattern = re.compile(r"\ba\w*", re.IGNORECASE)
pos = 0
while True:
  m = pattern.search(text, pos)
  if m:
    print(m.group(), m.span())
    pos = m.end() + 1
  else:
    break
```

- **Χωρισμός της συμβολοσειράς σε μέρη (διαχωρισμός με βάση την κανονική έκφραση):**

Συνάρτηση/Μέθοδος	Επεξήγηση
split(pattern, text, maxsplits=0, flags=0)	Χωρίζει τις συμβολοσειρές με διαχωριστή εμφανίσεις του pattern maxsplits: αν δεν είναι μηδέν, τότε θα κάνει το πολύ maxsplits χωρισμούς (ό,τι απομένει θα πάει στην τελευταία συμβολοσειρά)
Pattern.split (text, maxsplit=0)	Ομοίως, χρησιμοποιώντας τη κ.ε. που έχει γίνει πριν compile.

- Επιστρέφει λίστα με τις συμβολοσειρές που απομένουν έχοντας αφαιρέσει τους διαχωριστές που προέκυψαν από την κανονική έκφραση.

Παράδειγμα 5: splitting

```
import re

text = "Computer Science is no more about computers, " + \
    "than astronomy is about telescope."

print(re.split(r" ", text))
print(re.split(r"\W? \W?", text))
print(re.split(r"\W? +\W?|\W", text))
```

Εντοπισμός όλων των υποσυμβολοσειρών που ταιριάζουν με μία κανονική έκφραση μίας συμβολοσειράς:

Συνάρτηση/Μέθοδος	Ταίριασμα με:
findall(pattern, text, flags)	Εντοπίζει τις εμφανίσεις του pattern μέσα στη συμβολοσειρά text. Τα τμήματα που επιστρέφει δεν είναι αλληλοεπικαλυπτόμενα.
Pattern.findall(text [start[, end]])	Ομοίως με έξτρα ορίσματα το διάστημα του text στο οποίο θα γίνει η αναζήτηση

- επιστρέφουν λίστα με τα ταιριάσματα που έγιναν.

Παράδειγμα 6: findall.py

```
text = "..."

print(re.findall(r"\w+", text))
print(re.findall(r"(\w+) about (\w+)", text))
print(re.findall(r"((\w+) about (\w+))", text))

text = ... +  "and so is everything about everybody else.\n " + ...
pattern = re.compile(r"^(Y\w+) ([\w']+) (\w+)", re.MULTILINE)
print(pattern.findall(text))
```

- Επίσης:
  - Η συνάρτηση **finditer**(pattern, text, flags) και η
  - Η μέθοδος Pattern.**finditer**(text, [start[, end]])
- κάνουν τα ίδια με την findall αλλά επιστρέφουν iterator (αντί για λίστα)

- **Αντικατάσταση ταιριασμάτων με άλλες συμβολοσειρές**

Συνάρτηση/Μέθοδος	Επεξήγηση
sub(pattern, repl, text, count=0, flags=0)	Εντοπίζει τις εμφανίσεις της pattern στην text και τις αντικαθιστά με τη repl
Pattern.sub(repl, text, count=0)	Ομοίως

- Επιστρέφει τη συμβολοσειρά που προκύπτει μετά από την αντικατάσταση.
- count: Το πλήθος των αντικαταστάσεων που θα γίνουν (αν =0, τότε θα γίνουν όλες οι αντικαταστάσεις)

Παράδειγμα 7: sub

```
text = "..."

res = re.sub(r"the\w*", "---", text)
print(res)

with open("pies.html", "r", encoding="utf-8") as f:
    text = f.read()
    s = re.sub(r"<ul>", "<ol>", text, 0, re.MULTILINE)
    s2 = re.sub(r"</ul>", "</ol>", s, 0, re.MULTILINE)
with open("pies2.html", "w", encoding="utf-8") as f:
    f.write(s2)
```

Σημείωση:

- Το repl (όρισμα της sub) μπορεί να είναι ακόμη και μία συνάρτηση:
  - πρέπει να παίρνει σαν όρισμα ένα αντικείμενο τύπου match (αυτό με το οποίο έγινε το ταίριασμα)
  - Επιστρέφει τη συμβολοσειρά με την οποία γίνεται η αντικατάσταση.

Παρεμφερείς είναι και οι:

Συνάρτηση/Μέθοδος	Ταίριασμα με:
subn(pattern, repl, text, count=0, flags=0)	ίδια με τη sub(), αλλά επιστρέφει tuple με δύο στοιχεία: τη συμβολοσειρά και τις αντικαταστάσεις που έγιναν.
Pattern.subn(repl, text, count=0)	Ομοίως

Παράδειγμα 8: subn.py

```
text = "Men occasionally stumble over the truth, but most of them " \
       "pick themselves up and hurry off as if nothing had happened."

def replace(match_object):
    length = len(match_object.group(0))
    return "-" * length

res = re.subn(r"the\w*", replace, text)
print(res)
```