**Laboratorium 10 (Programowanie 2 – 410-KS1-2PRO3)**
**Kierunek kognitywistyka i komunikacja**
**Wyrażenia regularne cd.**

1. **(jeszcze raz polecenia regex)**
   **Metaznaki to znaki, które są interpretowane w specjalny sposób przez silnik RegEx. Oto lista metaznaków: [] . ^ $ * + ? {} () \ |**
   - **[] - Nawiasy kwadratowe określają zestaw znaków, które chcemy dopasować**
   - **. - Kropka pasuje do dowolnego pojedynczego znaku (z wyjątkiem znaku nowej linii „\n")**
   - **^ - Daszek służy do sprawdzania, czy ciąg zaczyna się od określonego znaku.**
   - **$ - Dolar służy do sprawdzenia, czy ciąg kończy się określonym znakiem.**
   - *** - Gwiazdka dopasowuje zero lub więcej wystąpień wzorca.**
   - **+ - Plus dopasowuje jedno lub więcej wystąpień wzorca.**
   - **? - Znak zapytania dopasowuje zero lub jedno wystąpienie wzorca.**
   - **{} - Nawiasy klamrowe {*n,m*} oznacza to co najmniej *n*, a co najwyżej *m* powtórzeń wzoru pozostawionego temu.**
   - **| - Alternatywne znaki/wzorca (operator *lub*).**
   - **() — Grupa służy do grupowania podwzorów.**
   - **\ - Odwrotny ukośnik służy do poprzedzania różnych znaków, w tym wszystkich metaznaków.**

   - **\A — Dopasowuje, jeśli wzorzec znajduje się na początku ciągu.**
   - **\b — Dopasowuje, jeśli wzorzec znajduje się na początku lub na końcu słowa.**
   - **\B — Przeciwny do \b. Dopasowuje, jeśli określone znaki nie znajdują się na początku ani na końcu słowa**
   - **\d — Dopasowuje do dowolnej cyfry dziesiętnej. Odpowiednik [0-9]**
   - **\D — Dopasowuje do dowolnej cyfry niedziesiętnej. Odpowiednik [^0-9]**
   - **\s — Dopasowuje, gdy ciąg zawiera dowolny biały znak. Odpowiednik [ \t\n\r\f\v]**
   - **\S — Dopasowuje, gdy ciąg zawiera dowolny znak niebędący biały znakiem. Odpowiednik [^ \t\n\r\f\v]**
   - **\w — Dopasowuje dowolny znak alfanumeryczny (cyfry i alfabety). Odpowiednik [a-zA-Z0-9_]**
   - **\W — Dopasowuje do dowolnego znaku niealfanumerycznego. Odpowiednik [^a-zA-Z0-9_]**
   - **\Z — Dopasowuje koniec ciągu. Jeśli istnieje znak nowej linii, dopasowuje się tuż przed znakiem nowej linii.**
   - **\z — Dopasowuje koniec ciągu.**
   - **\G - Punkt dopasowania, w którym zakończyło się ostatnie dopasowanie.**
   - **\n, \t, itp. — Dopasowuje znaki nowej linii, powrót karetki, tabulatory itp.**
   - **\1...\9 — Dopasowuje *n*-tą grupę podwyrażenie.**

- **\10 — Dopasowuje do *n*-tego zgrupowanego podwyrażenia, jeśli taka grupa wystąpiła. W przeciwnym razie odnosi się do ósemkowej reprezentacji kodu znaku.**
-

2. **(pakiet *re* polecenia)**

```
# re.match()
print(re.match.__doc__)
string = "Python is fun"
match = re.match('Python', string)
print(match.group())
print(match.start())
print(match.end())
```

```
# re.search()
print(re.search.__doc__)
string = "Python is fun"
match = re.search('\APython', string)
if match:
  print("pattern found inside the string")
else:
  print("pattern not found")

string = '39801 356, 2102 1111'
pattern = '(\d{3}) (\d{2})'
match = re.search(pattern, string)
if match:
  print(match.group())
else:
  print("pattern not found")

print(match.group(1))
print(match.group(2))
print(match.group(1, 2))
print(match.start())
print(match.end())
print(match.span())
print(match.re)
print(match.string)
```

```
# re.findall()
print(re.findall.__doc__)

string = 'hello 12 hi 89. Howdy 34'
pattern = '\d+'

result = re.findall(pattern, string)
print(result)
```

```
# re.split()
print(re.split.__doc__)
string = 'Twelve:12 Eighty nine:89 Nine:9.'
pattern = '\d+'

result = re.split(pattern, string)
print(result)

result = re.split(pattern, string, maxsplit=1)
```

```
print(result)

result = re.split(pattern, string, maxsplit=2)
print(result)
```

```
# re.sub()
print(re.sub.__doc__)
string = 'abc 12\
de 23 \n f45 6'
pattern = '\s+'
replace = ''
new_string = re.sub(pattern, replace, string)
print(new_string)

new_string = re.sub(r'\s+', replace, string, count=1)
print(new_string)
new_string = re.sub(r'\s+', replace, string, count=2)
print(new string)

new_string = re.subn(pattern, replace, string)
print(new_string)
new_string = re.subn(pattern, replace, string, 1)
print(new_string)
```

| | | |
|---|---|---|
| ***match()*** | ***Dopasowanie wzorca na początku łańcucha.*** | ***Match*** |
| ***search()*** | ***Dopasowanie w dowolnym miejscu łańcucha.*** | ***Match*** |
| ***findall()*** | ***Dopasowanie wszystkich, niezachodzących dopasowań.*** | ***lista*** |
| ***finditer()*** | ***Dopasowanie wszystkich, niezachodzących dopasowań.*** | ***iterator*** |
| ***split()*** | ***Pocięcie łańcucha w miejscach dopasowań.*** | ***lista*** |
| ***sub()*** | ***Zamiana dopasowania wzorca na inny łańcuch.*** | ***Łańcuch znaków*** |

3. **(falgi w pakiecie re)**
   - **re.I - Wykonuje dopasowanie bez rozróżniania wielkości liter.**
   - **re.L - Interpretuje słowa zgodnie z aktualną lokalizacją. Ta interpretacja wpływa na grupę alfabetyczną (\w i \W), a także na zachowanie granic wyrazów (\b i \B).**
   - **re.M - Sprawia, że $ pasuje do końca linii (nie tylko do końca łańcucha), a ^ dopasowuje początek dowolnej linii (nie tylko początek łańcucha). odp. Sprawia, że kropka (kropka) pasuje do dowolnego znaku, w tym do znaku nowej linii.**
   - **re.U - Interpretuje litery zgodnie z zestawem znaków Unicode. Ta flaga wpływa na zachowanie \w, \W, \b, \B.**
   - **re.X - Pozwala na „ładniejszą" składnię wyrażeń regularnych. Ignoruje białe znaki (z wyjątkiem znaków znajdujących się wewnątrz zestawu [] lub po opuszczeniu przez odwrotny ukośnik) i traktuje niewymienione znak # jako znacznik komentarza.**
4. **(polecenia *regex*) Zapoznaj się z następującymi symbolami stosowanymi w tworzeniu wyrażeń regularnych**

| | |
|---|---|
| ***wzorzec(?=X)*** | ***po wzorcu powinien występować X*** |
| ***wzorzec(?!X)*** | ***po wzorcu nie powinien występować X*** |
| ***(?<=X)wyrażenie*** | ***X poprzedza wyrażenie.*** |
| ***(?<!X)wyrażenie*** | ***X nie poprzedza wyrażenia.*** |

**Dopasowania z użyciem znaków * oraz + jest zachłanne, co oznacza, że dopasowuje tak wiele znaków, jak to tylko możliwe. Dodanie znaku ?, zmienia to zachowanie, na leniwe - dopasowane jest tak mało znaków, jak to możliwe.**

**5.**