Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71230985**
Nama Lengkap	**TOMAS BECKET**
Minggu ke / Materi	**14 / Regular Expresion**

SAYA MENYATAKAN BAHWA LAPORAN PRAKTIKUM INI SAYA BUAT DENGAN USAHA SENDIRI TANPA MENGGUNAKAN BANTUAN ORANG LAIN. SEMUA MATERI YANG SAYA AMBIL DARI SUMBER LAIN SUDAH SAYA CANTUMKAN SUMBERNYA DAN TELAH SAYA TULIS ULANG DENGAN BAHASA SAYA SENDIRI.

SAYA SANGGUP MENERIMA SANKSI JIKA MELAKUKAN KEGIATAN PLAGIASI, TERMASUK SANKSI TIDAK LULUS MATA KULIAH INI.

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN DUTA WACANA
YOGYAKARTA
2024

BAGIAN 1: MATERI MINGGU INI (40%)

MATERI 1

Regular Expression atau biasa disebut RegEx adalah sebuah pola yang digunakan untuk mencocokkan sekumpulan string dalam proses pencarian teks. RegEx digunakan secara luas dalam pemrograman dan pengolahan teks untuk berbagai tujuan, seperti pencarian, penggantian, dan validasi teks.

RegEx sangat powerful dalam proses searching dan extracting pola dalam teks. Misalnya, kita dapat menggunakan RegEx untuk menemukan semua alamat email dalam sebuah dokumen, mengganti semua nomor telepon dengan format yang lebih konsisten, atau bahkan menghapus semua tag HTML dari sebuah halaman web. Meskipun demikian, salah satu kekurangan utama dari RegEx adalah pola-pola yang digunakannya bisa sangat rumit dan sulit untuk dibaca serta dipahami, terutama bagi pemula.

Tidak semua bahasa pemrograman mendukung penggunaan RegEx secara native. Namun, banyak bahasa pemrograman modern yang menyediakan dukungan untuk RegEx melalui berbagai library atau modul. Python adalah salah satu bahasa pemrograman yang mendukung library RegEx dengan sangat baik. Di Python, kita dapat menggunakan library re untuk bekerja dengan RegEx.

Library re di Python menyediakan berbagai fungsi yang dapat digunakan untuk mencari, mengganti, dan memanipulasi string berdasarkan pola RegEx. Salah satu fungsi yang paling mudah digunakan dari library re adalah search(). Fungsi search() memungkinkan kita untuk mencari pola tertentu dalam sebuah string dan mengembalikan objek match jika pola ditemukan. Jika tidak, fungsi ini akan mengembalikan None.

Contoh seperti source code dibawah :

```
import re

handle=open('mbox-short.txt')
count = 0
for line in handle:
    line=line.rstrip()
    if re.search('From:', line):
        count += 1
        rint(line)
    print("Count: ",count)
```

dapat melihat bahwa re.search bisa saja diganti dengan menggunakan perintah find() pada string biasa.

MATERI 2

PENGGUNAAN FINDALL

Fungsi findall bergantung pada program atau library yang digunakan, tetapi umumnya digunakan untuk mencari semua kemunculan suatu pola tertentu dalam teks atau string dan findall mengambil dua argumen:

1. **Pola pencarian**: Ini bisa berupa string biasa atau ekspresi regular (regex) yang menentukan pola yang ingin Anda temukan.
2. **Teks atau string**: Ini adalah teks atau string di mana Anda akan mencari pola tersebut.

findall kemudian akan mencari semua kemunculan pola dalam teks atau string tersebut dan mengembalikan daftar semua kecocokan yang ditemukan.

Contoh seperti Source Code dibawah :

```
import re

txt = "The rain in Spain 12:00 AM"

y = re.findall('[a-zA-Z]', txt)
x = re.findall('[0-9][0-9]:[0-9][0-9]', txt)
print(y)
print(x)
```

jadi pada baris y = re.findall('[a-zA-Z]', txt), aris ini menggunakan fungsi re.findall() untuk mencari semua huruf besar dan kecil dalam string txt. Pola [a-zA-Z] mencocokkan setiap huruf dari 'a' hingga 'z' dan dari 'A' hingga 'Z'. Fungsi findall mengembalikan daftar semua kecocokan yang ditemukan. Dalam hal ini, y akan berisi daftar semua huruf dalam teks.

Sedangkan pada Baris x = re.findall('[0-9][0-9]:[0-9][0-9]', txt), menggunakan fungsi re.findall() untuk mencari pola waktu dalam format HH (dua digit angka, diikuti oleh titik dua, lalu dua digit angka) dalam string txt. Pola [0-9][0-9]:[0-9][0-9] mencocokkan setiap waktu dalam format tersebut. Fungsi findall mengembalikan daftar semua kecocokan yang ditemukan. Dalam hal ini, x akan berisi daftar waktu yang ditemukan dalam teks.

Sehingga jika kita lihat hasil outpunya maka akan seperti :

Print(y) #OUTPUT ['T', 'h', 'e', 'r', 'a', 'i', 'n', 'i', 'n', 'S', 'p', 'a', 'i', 'n', 'A', 'M']

Print(x) #OUTPUT ['12:00']

MATERI 3

CONTOH LAIN PENGGUNAAN FINDALL

Source Code:

```
import re

iniString = "Sang mata-mata g memata-matai mata"
iniString = re.findall("^S\w.g",iniString)
print(iniString)
```

Pada baris iniString = re.findall("^S\w.g",iniString), Baris ini menggunakan fungsi re.findall() untuk mencari pola dalam string iniString. Pola yang digunakan adalah ^S\w.g. Mari kita uraikan pola ini:

- ^ : Menandakan bahwa pencarian harus dimulai dari awal string.
- S : Karakter S harus ada di awal string.
- \w : Karakter alfanumerik (huruf atau angka) atau underscore _.
- . : Karakter apa saja kecuali newline.
- g : Karakter g.

Jadi, pola ^S\w.g mencari substring yang dimulai dengan S, diikuti oleh satu karakter alfanumerik, diikuti oleh karakter apa saja, dan diakhiri dengan g, dan harus berada di awal string.

Ada banyak pola lainnya atau Special Character yang dimiliki :

Karakter	Kegunaan	Contoh	Arti Contoh
[]	Kumpulan karakter	"[a-zA-Z]"	1 karakter antara a-z kecil atau A-Z besar
\{}	Karakter dengan arti khusus dan escaped character	\{}d	Angka / digit
.	Karakter apapun kecuali newline	say.n.	Tidak bisa diganti dengan karakter apapun, misal "sayang" akan valid
^	Diawali dengan	^From	Diawali dengan From
$	Dakhiri dengan	this$	Diakhiri dengan kata this
*	0 s/d tak terhingga karakter	\{}d*	ada digit minimal 0 maksimal tak terhingga
?	ada atau tidak (opsional)	\{}d?	Boleh ada atau tidak ada digit sebanyak
+	1 s/d tak terhingga karakter	\{}d+	Minimal 1 s/d tak terhingga karakter
{}	Tepat sebanyak yang ada para {}	\{}d{2}	Ada tepat 2 digit
()	Pengelompokan karakter / pola	(sayalkamu)	saya atau kamu sebagai satu kesatuan
|	atau	\{}d|\{}s	1 digit atau 1 spasi

Dan masi banyak lagi,

Special Characters	Kegunaan	Contoh
\b	Digunakan untuk mengetahui apakah suatu pola berada di awal kata atau akhir kata	"R\bin" "Rain\b"
\d	Digunakan untuk mengetahui apakah karakter adalah sebuah digit (0 s/d 9)	\d
\D	Digunakan untuk mengetahui apakah karakter yang bukan digit	\D
\s	Digunakan untuk mengetahui apakah karakter adalah whitespace (spasi, tab, enter)	\s
\S	Digunakan untuk mengetahui apakah karakter adalah BUKAN whitespace (spasi, tab, enter)	\S
\w	Digunakan untuk mengetahui apakah karakter adalah word (a-z, A-Z, 0-9, dan _)	\w
\W	Digunakan untuk mengetahui apakah karakter adalah BUKAN word (a-z, A-Z, 0-9, dan _)	\W
\A	Digunakan untuk mengetahui apakah karakter adalah berada di bagian depan dari kalimat	"\AThe"
\Z	Digunakan untuk mengetahui apakah karakter adalah berada di bagian akhir dari kalimat	"End\Z"

Sehingga ketika code tersebut mencetak hasilnya maka ouputnya yang muncul yaitu **['Sang']**

Dan kesimpulannya **^S\w.g** akan mencocokkan substring di awal string yang dimulai dengan S, diikuti oleh satu karakter alfanumerik, diikuti oleh karakter apa saja, dan diakhiri dengan g.

Untuk string "Sang mata-mata g memata-matai mata":

- Pola ^S\w.g akan mencocokkan substring "Sang" karena "Sang" berada di awal string dan sesuai dengan pola.

Oleh karena itu, hasil dari pencarian adalah daftar yang berisi satu elemen: ['Sang'].

```
import re

iniString = "Sang mata-mata g memata-matai mata"
iniString = re.findall("^S\w.g", iniString)
print(iniString)  # Output: ['Sang']
```

MATERI 4

PENGGUNAAN SEARCH

re.search adalah fungsi dalam library re di Python yang digunakan untuk mencari pola dalam sebuah string. Fungsi ini memeriksa string untuk setiap kecocokan dengan pola yang ditentukan, dan mengembalikan objek match pertama yang ditemukan. Jika tidak ada kecocokan yang ditemukan, fungsi ini mengembalikan None.

Contoh Source Code :

```
import re

txt = "Sang mata-mata memata-matai mata"

iniString = "Sang mata-mata g memata-matai mata"
iniString = re.search("matai",iniString)

print(iniString.start())
print(iniString.group())
print(iniString.end())
```

Pada source code diatas, baris iniString = re.search("matai",iniString), akan mencari pola "matai" dalam string iniString. Fungsi re.search() mencari kemunculan pertama dari pola yang ditentukan dalam string. Jika pola ditemukan, fungsi ini mengembalikan objek match; jika tidak, mengembalikan None.

Dalam string "Sang mata-mata g memata-matai mata", pola "**matai**" ditemukan di kata "memata-matai".

Sedangkan pada print .group() Mengembalikan bagian string yang cocok dengan pola. Dalam contoh ini, group() akan mengembalikan "matai" karena itu adalah kecocokan yang ditemukan

dan print .end() Mengembalikan posisi akhir (indeks) dari kecocokan pertama. Dalam contoh ini, end() akan mengembalikan indeks 26 karena "matai" berakhir pada indeks ke-25 (posisi ke-26 jika dihitung dari 0).

Sehingga hasil outputnya :

```
24
matai
29
```

MATERI 5

PENGGUNAAN SUB

Fungsi **re.sub** dalam library re di Python digunakan untuk mengganti substring dalam sebuah string yang sesuai dengan pola tertentu dengan substring lain. Ini sangat berguna untuk berbagai tugas manipulasi teks seperti penggantian kata, pembersihan teks, atau format ulang teks.

Contoh Source Code :

```
import re

txt = "Sang mata-mata memata-matai mata"
```

```
iniString = "Sang mata-mata g memata-matai mata"
iniString = re.sub("mata", "hidung",iniString)

print(iniString)
```

misal pada baris iniString = re.sub("mata", "hidung",iniString), akan mengganti semua kemunculan pola "mata" dalam string iniString dengan substring "hidung". Fungsi re.sub() mencari semua kecocokan pola "mata" dan menggantinya dengan "hidung". Sehingga ouput yang munucl yaitu **Sang hidung-hidung g mehidung-hidungi hidung**

MATERI 6

PENGGUNAAN SPLIT

Fungsi dari re.split sendiri digunakan untuk memisahkan substring pada sebuah string yang sekiranya sesuai dengan pola yang ingin dicari, misal pada Source code dibawah :

```
import re

txt = "Sang mata-mata memata-matai mata"

iniString = "Sang mata-mata g memata-matai mata"
iniString = re.split("[ma]",iniString)

print(iniString)
```

Baris iniString = re.split("[ma]",iniString), akan mencari pola string yang memiliki huruf ma sehingga ketika huruf ini ditemukan maka semua kalimat akan menambahkan spasi ketika kata pada string memiliki pola ma

Outputnya yaitu :

```
['S', 'ng ', '', 't', '-', '', 't', ' g ', 'e', '', 't', '-', '', 't', 'i ', '', 't', '']
```

LINK GITHUB : https://github.com/TomasBeckett/PrakAlpro15.git

BAGIAN 2: LATIHAN MANDIRI (60%)

SOAL 1

1.1 SOURCE CODE

```
import re
from datetime import datetime

teks = """
Pada tanggal 1945-08-17 Indonesia merdeka. Indonesia memiliki beberapa pahlawan
nasional, seperti Pangeran Diponegoro (TL: 1785-11-11), Pattimura (TL: 1783-06-
08) dan Ki Hajar Dewantara (1889-05-02).
"""

pola = r'\d{4}-\d{2}-\d{2}'

tanggal_list = re.findall(pola, teks)

tanggal_sekarang = datetime.now()

for tanggal_str in tanggal_list:
    tanggal_obj = datetime.strptime(tanggal_str, "%Y-%m-%d")

    delta_hari = (tanggal_sekarang - tanggal_obj).days

    formatted_tanggal = tanggal_obj.strftime("%Y-%m-%d %H:%M:%S")
    result = f"{formatted_tanggal} selisih {delta_hari} hari"
    print(result)
```

1.2 PENJELASAN

```
pola = r'\d{4}-\d{2}-\d{2}'
```

pola pada line tersebut menggunakan pola regular expression yang digunakan untuk mencari format tanggal YYYY-MM-DD dalam teks. Penjelasan polanya yaitu:

- \d{4}: Mencocokkan 4 digit angka untuk tahun.
- -: Mencocokkan tanda penghubung "-" setelah tahun.
- \d{2}: Mencocokkan 2 digit angka untuk bulan.
- -: Mencocokkan tanda penghubung "-" setelah bulan.
- \d{2}: Mencocokkan 2 digit angka untuk hari.

```
tanggal_list = re.findall(pola, teks)
```
dan Baris ini menggunakan re.findall untuk menemukan semua kemunculan tanggal dalam format YYYY-MM-DD dalam teks dan menyimpannya dalam tanggal_list

```
tanggal_sekarang = datetime.now()
```
sedangkan pada Baris ini akan mendapatkan tanggal dan waktu saat ini menggunakan fungsi datetime.now() dan menyimpannya dalam variabel tanggal_sekarang.

```
for tanggal_str in tanggal_list:
    tanggal_obj = datetime.strptime(tanggal_str, "%Y-%m-%d")

    delta_hari = (tanggal_sekarang - tanggal_obj).days

    formatted_tanggal = tanggal_obj.strftime("%Y-%m-%d %H:%M:%S")
    result = f"{formatted_tanggal} selisih {delta_hari} hari"
```
Setelah mendapatkan daftar tanggal dalam format yang benar, program melakukan loop untuk setiap tanggal dalam daftar tersebut.

- datetime.strptime(tanggal_str, "%Y-%m-%d"): Baris ini mengonversi string tanggal dalam format YYYY-MM-DD menjadi objek datetime.
- (tanggal_sekarang - tanggal_obj).days: Baris ini menghitung selisih hari antara tanggal sekarang dan tanggal yang ditemukan.
- tanggal_obj.strftime("%Y-%m-%d %H:%M:%S"): Baris ini mengonversi objek datetime kembali menjadi string dengan format YYYY-MM-DD HH:MM:SS.
- result = f"{formatted_tanggal} selisih {delta_hari} hari": Baris ini membuat string hasil dengan menambahkan tanggal dalam format yang diformat dan selisih hari.

1.3 HASIL OUTPUT

```
1945-08-17 00:00:00 selisih 28780 hari
1785-11-11 00:00:00 selisih 87132 hari
1783-06-08 00:00:00 selisih 88019 hari
1889-05-02 00:00:00 selisih 49340 hari
```

SOAL 2

2.1 SOURCE CODE

```
import re
import random
import string

emails = """
anton@mail.com dimiliki oleh antonius
budi@gmail.co.id dimiliki oleh budi anwari
slamet@getnada.com dimiliki oleh slamet slumut
matahari@tokopedia.com dimiliki oleh toko matahari
"""

email_pattern = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'

email_list = re.findall(email_pattern, emails)

for email in email_list:
    username = email.split('@')[0]
    password = ''.join(random.choices(string.ascii_letters + string.digits, k=8))
    print(f"{email} username: {username} , password: {password}")
```

2.2 PENJELASAN

```
email_pattern = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
```

pola pada line tersebut sama seperti latihan1 yaitu meggunakan regular expresion yang digunakan untuk mencocokkan format email dalam teks. Penjelasan polanya yaitu:

- \b: Batas kata.
- [A-Za-z0-9._%+-]+: Karakter-karakter yang diperbolehkan sebelum simbol "@" dalam email.
- @: Simbol "@".
- [A-Za-z0-9.-]+: Karakter-karakter yang diperbolehkan setelah simbol "@" dalam domain.
- \.: Simbol "." yang harus ada setelah domain.
- [A-Z|a-z]{2,}: Domain harus memiliki minimal 2 karakter alfabet.
- \b: Batas kata.

```
email_list = re.findall(email_pattern, emails)
```

pada Baris ini menggunakan re.findall untuk menemukan semua email yang cocok dengan pola regex dari teks emails.

```
for email in email_list:
    username = email.split('@')[0]
    password = ''.join(random.choices(string.ascii_letters + string.digits, k=8))
    print(f"{email} username: {username} , password: {password}")
```

Loop for email in email_list:: Program melakukan loop untuk setiap email yang ditemukan.

- username = email.split('@')[0]: Baris ini memisahkan username dari email dengan menggunakan "@" sebagai pemisah dan mengambil bagian pertama sebelum "@".
- password = ''.join(random.choices(string.ascii_letters + string.digits, k=8)): Baris ini menghasilkan password acak 8 karakter yang terdiri dari huruf dan angka.
- print(f"{email} username: {username} , password: {password}"): Baris ini mencetak hasil dengan menampilkan email, username, dan password yang dihasilkan.

Sehingga kode tersebut mencari dan memproses email dalam teks, kemudian menghasilkan username dari email dan password acak 8 karakter untuk setiap email yang ditemukan.

2.3 HASIL OUTPUT

```
anton@mail.com username: anton , password: ozFWY02I
budi@gmail.co.id username: budi , password: 5tAb0y8H
slamet@getnada.com username: slamet , password: 8QZTMhWE
matahari@tokopedia.com username: matahari , password: C8Izmle6
```

LINK GITHUB : https://github.com/TomasBeckett/PrakAlpro15.git