



Laporan Praktikum Algoritma & Pemrograman

Semester Genap 2024/2025

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.

NIM	<71231046>
Nama Lengkap	<FREIRE HANAN PUTRA >
Minggu ke / Materi	08 / Pengolahan String dan Regular Expression

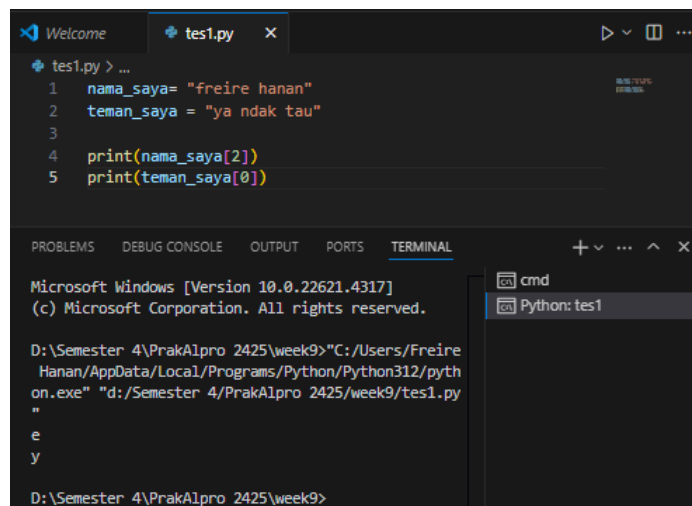
PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS KRISTEN DUTA WACANA
YOGYAKARTA
2025

BAGIAN 1: MATERI MINGGU INI (40%)

MATERI 1 Pengantar String

String adalah rangkaian karakter yang digabung menjadi satu kesatuan dan digunakan dalam program komputer untuk menyimpan teks, baik yang pendek maupun panjang. String merupakan jenis data yang dapat menampung huruf atau karakter, dan biasanya disimpan dalam format kode ASCII. Tidak semua bahasa pemrograman menyediakan tipe data String secara langsung, contohnya bahasa C. String bukan termasuk tipe data dasar karena menyimpan lebih dari satu nilai dalam satu struktur. Di beberapa bahasa pemrograman, String dianggap sebagai kumpulan karakter, baik berupa array maupun daftar karakter.

MATERI 2 Pengaksesan String dan Pengolahan String



```
tes1.py > ...
1  nama_saya= "freire hanan"
2  teman_saya = "ya ndak tau"
3
4  print(nama_saya[2])
5  print(teman_saya[0])

PROBLEMS  DEBUG CONSOLE  OUTPUT  PORTS  TERMINAL

Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

D:\Semester 4\PrakAlpro 2425\week9>"C:/Users/Freire
Hanan/AppData/Local/Programs/Python/Python312/pyth
on.exe" "d:/Semester 4/PrakAlpro 2425/week9/tes1.py"
"
e
y
D:\Semester 4\PrakAlpro 2425\week9>
```

String pertama kali dibuat dengan cara mendeklarasikan variabel sekaligus mengisinya dengan data. String bisa diakses secara keseluruhan cukup dengan menyebut nama variabel, atau bisa juga diakses per karakter dengan menggunakan indeks. Indeks pada string dimulai dari angka 0, mirip seperti pada struktur data list. Perlu diingat bahwa indeks yang digunakan harus berupa bilangan bulat, bukan desimal. Di dalam memori komputer, string disimpan secara berurutan dalam bentuk list yang berisi karakter-karakter dengan indeks yang dimulai dari nol.

b	a	n	a	n	a
[0]	[1]	[2]	[3]	[4]	[5]

MATERI 3 Operator dan Metode String

Operator in

Di dalam string itu dapat memeriksa apakah dalam sebuah kalimat merupakan substring dari sebuah kalimat lain menggunakan operator in. hasil dari pemeriksaan tersebut adalah True/Falase

```

1 kalimat = "saya sudah makan"
2 data = "mau"
3
4 print(kalimat in data) #false
5 print("mau" in kalimat) #false

```

Selain operator `in` pada string juga dapat melakukan comparison atau perbandingan yang juga menghasilkan True or False

```

1 if "saya" > "dia":
2     print("ya")
3 else:
4     print("tidak")
5
6 if "dua" == "dua":
7     print("sama")

```

Fungsi Len

Untuk mengetahui jumlah karakter dalam sebuah string, gunakan fungsi `len(<string>)`. Untuk mengambil huruf terakhir, akses indeks ke `len(<string>) - 1` karena indeks dimulai dari 0.

```

1 kalimat = "universitas kristen duta wacana yogyakarta"
2 print(len(kalimat)) #output 42
3 terakhir = kalimat[len(kalimat)-1]
4 print(terakhir) #output 'a'
5
6 #bisa juga menggunakan indeks -1
7 terakhir_versi2 = kalimat[-1]
8 print(terakhir_versi2) #output 'a'
9 #atau menggunakan indeks -2 untuk huruf terakhir kedua
10 terakhir2 = kalimat[-2]
11 print(terakhir2) #output 't'

```

Traversing String

Untuk dapat menampilkan string dengan cara ditampilkan huruf demi huruf adalah dengan menggunakan loop yang dilakukan per huruf dengan 2 cara:

- Dengan cara akses terhadap indeks

```

1 kalimat = "indonesia jaya"
2
3 i = 0
4 while i < len(kalimat):
5     print(kalimat[i], end=" ")
6     i += 1
7
8 # output: indonesia jaya

```

- Dilakukan tanpa akses terhadap indeks otomatis

```

1 kalimat = "indonesia jaya"
2
3 for kal in kalimat:
4     print(kal, end="")
5
6 # output: indonesia jaya

```

String Slivce

String slice adalah teknik untuk mengambil sebagian dari string (substring) dengan menentukan indeks mulai dan indeks akhir-1. Penulisannya menggunakan format <string>[awal:akhir], di mana bagian awal atau akhir bisa dikosongkan. Perhitungan indeks awal dimulai dari 0.

```

1 kalimat = "cerita rakyat"
2 awal = 0
3 akhir = 6
4 print(kalimat[awal:akhir]) #cerita
5 print(kalimat[7:len(kalimat)]) #rakyat
6 print(kalimat[:5]) #cerit
7 print(kalimat[5:]) #a rakyat
8 print(kalimat[:]) #cerita rakyat

```

String merupakan data yang bersifat immutable! Immutable adalah bahwa data tersebut tidak bisa diubah saat program berjalan, hanya bisa diinisialisasi saja

Di bawah ini merupakan table method String yang biasa digunakan, selain yang ada di table bawah ini dapat diakses di <https://docs.python.org/library/stdtypes.html#string-methods> :

Nama Method	Kegunaan	Penggunaan
capitalize()	untuk mengubah string menjadi huruf besar	string.capitalize()
count()	menghitung jumlah substring yang muncul dari sebuah string	string.count()
endswith()	mengetahui apakah suatu string diakhiri dengan string yang diinputkan	string.endswith()
startswith()	mengetahui apakah suatu string diawali dengan string yang diinputkan	string.startswith()
find()	mengembalikan indeks pertama string jika ditemukan string yang dicari	string.find()
islower() dan isupper()	mengembalikan True jika string adalah huruf kecil / huruf besar	string.islower() dan string.isupper()
isdigit()	mengembalikan True jika string adalah digit (angka)	string.isdigit()
strip()	menghapus semua whitespace yang ada di depan dan di akhir string	string.strip()
split()	memecah string menjadi token-token berdasarkan pemisah, misalnya berdasarkan spasi	string.split()

Dalam Python, operator + dapat digunakan untuk menggabungkan dua string, sedangkan operator * bisa digunakan untuk mengulang string sesuai jumlah perkaliannya. Contohnya seperti dibawah ini:

```

1 kata1 = "saya"
2 kata2 = "makan"
3 kata3 = kata1 + " " + kata2
4 print(kata3) #hasil adalah penggabungan: saya makan
5 kata4 = "ulang"
6 print(kata4 * 4) #hasil adalah ulangulangulangulang
7 kata4 = "ulang "
8 print(kata4 * 2) #hasil adalah ulang ulang

```

MATERI 4 Parsing String

Parsing string adalah proses menelusuri dan memanipulasi bagian-bagian string untuk menemukan atau mengubah data tertentu. Misalnya, dari kalimat "Saudara-saudara, pada tanggal 17-08-1945 Indonesia merdeka", kita bisa mengambil tanggal dengan membagi string berdasarkan spasi, mencari bagian yang berisi angka, memecahnya dengan tanda '-' lalu menyusun ulang menjadi format 08/17/1945.

```

1 kalimat = "Saudara-saudara, pada tanggal 17-08-1945 Indonesia merdeka"
2 hasil = kalimat.split(" ")
3 for kal in hasil:
4     if kal[0].isdigit():
5         hasil2 = kal.split("-")
6         print(hasil2[1]+"-"+hasil2[0]+"-"+hasil2[2]) # output: 08/17/1945

```

MATERI 5 Pengantar RegEx

Dalam bab String, kita telah belajar berbagai teknik pengolahan string, termasuk dari file, namun teknik standar sering terasa rumit. Untuk itu, regular expression (regex) digunakan untuk mempermudah pencarian, penggantian, atau penghapusan string berdasarkan pola tertentu. Regex sangat kuat dalam mencari dan mengambil pola, meskipun sintaksnya cukup kompleks. Tidak semua bahasa pemrograman mendukung regex, tetapi Python mendukungnya melalui library re, dengan fungsi seperti search(). Sebagai contoh, kita bisa mencari semua string yang mengandung pola "From: " di file mbox-short.txt.

```

1 import re
2 handle=open('mbox-short.txt')
3 count = 0
4 for line in handle:
5     line=line.rstrip()
6     if re.search('From:', line):
7         count += 1
8     print(line)
9     print("Count: ",count)

```

Dari contoh diatas, sebenarnya re.search bisa digantikan dengan find() biasa, karena pola yang digunakan belum memanfaatkan kekuatan penuh regex. Namun jika dingin mencari baris diawali dengan pola "from", maka kita harus mengubah parameter fungsi search pada re.search menjadi re.search("from").

```

1 import re
2 handle=open('mbox-short.txt')
3 count = 0
4 for line in handle:
5     line=line.rstrip()
6     if re.search('^From:', line):
7         count += 1
8     print(line)
9     print("Count: ",count)

```

MATERI 6 Meta Character, Escaped Character, Set of Character, dan Fungsi Regex pada Library Python

Sebelum menggunakan fungsi regex perlu diketahui terlebih dahulu meta character/specia; character dan kegunaanya pada pola regex seperti pada table dibawah ini:

Tabel 8.2: Escaped Character pada Regex

Special Characters	Kegunaan	Contoh
\b	Digunakan untuk mengetahui apakah suatu pola berada di awal kata atau akhir kata	"R\bin" "Ra-in\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"

Tabel 8.1: Special Character pada Python

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

Dalam Python, terdapat beberapa karakter spesial (escaped characters) yang dijelaskan pada tabel 8.2. Selain itu, Python juga memungkinkan penggunaan himpunan karakter dengan simbol [], seperti yang ditunjukkan pada tabel 8.3.

Tabel 8.3: Himpunan Karakter pada Regex

[abc]	Mencari pola 1 huruf a, atau b, atau c
[a-c]	Mencari pola 1 huruf a s/d c
[^bmx]	Mencari pola 1 huruf yang bukan b,m, atau x
[012]	Mencari pola 1 huruf 0, atau 1, atau 2
[0-3]	Mencari pola 1 huruf 0 s/d 3
[0-2][1-3]	Mencari pola 2 huruf: 01, 02, 03, 11, 12, 13, 21, 22, 23
[a-zA-Z]	Mencari pola 1 huruf a-Z

Pada Python terdapat 4 buah fungsi yang bisa dipakai untuk menggunakan Regex seperti pada tabel 8.4

Tabel 8.4: Fungsi Regex pada Python

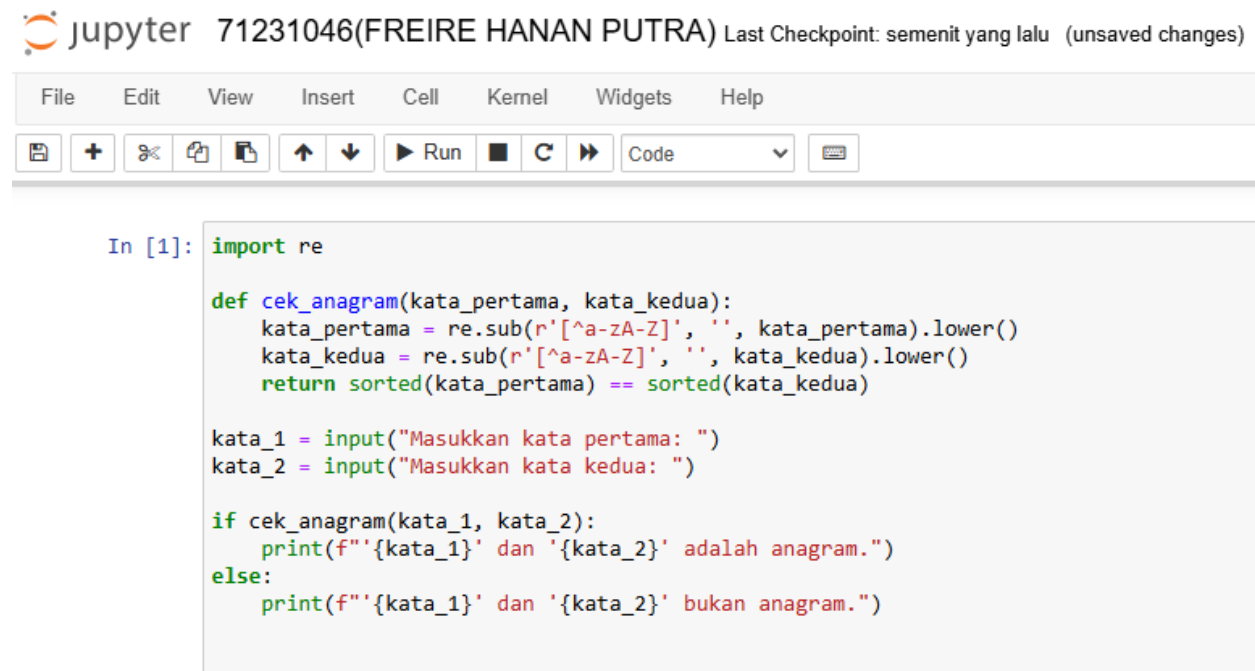
Nama Fungsi	Kegunaan
findall	mengembalikan semua string yang sesuai pola (matches)
search	mengembalikan string yang sesuai pola (match)
split	memecah string sesuai pola
sub	mengganti string sesuai dengan pola yang cocok

BAGIAN 2: LATIHAN MANDIRI (60%)

Link Github: https://github.com/Freirehnn23/prak_alpro_week8

SOAL 1

Code:



```
In [1]: import re

def cek_anagram(kata_pertama, kata_kedua):
    kata_pertama = re.sub(r'^a-zA-Z', '', kata_pertama).lower()
    kata_kedua = re.sub(r'^a-zA-Z', '', kata_kedua).lower()
    return sorted(kata_pertama) == sorted(kata_kedua)

kata_1 = input("Masukkan kata pertama: ")
kata_2 = input("Masukkan kata kedua: ")

if cek_anagram(kata_1, kata_2):
    print(f'{kata_1} dan {kata_2} adalah anagram.')
else:
    print(f'{kata_1} dan {kata_2} bukan anagram.')
```

Penerapan:

```
Masukkan kata pertama: kasur
Masukkan kata kedua: rusak
'kasur' dan 'rusak' adalah anagram.
```

Penjelasan:

1. Import Modul re:

- Modul re digunakan untuk manipulasi string menggunakan ekspresi reguler, dalam hal ini untuk menghapus karakter non-huruf.

2. Fungsi `cek_anagram(kata_pertama, kata_kedua)`:

- Menghapus Karakter Non-Huruf:**
 - `re.sub(r'^a-zA-Z', '', kata_pertama)` menghapus semua karakter yang bukan huruf dari kata_pertama.
 - `re.sub(r'^a-zA-Z', '', kata_kedua)` melakukan hal yang sama untuk kata_kedua.
- Mengubah ke Huruf Kecil:**

- `.lower()` digunakan untuk mengubah kedua kata menjadi huruf kecil, agar pencocokan tidak peka terhadap kapitalisasi.
- **Mengurutkan Karakter:**
 - `sorted(kata_pertama)` dan `sorted(kata_kedua)` mengurutkan karakter-karakter dalam kata pertama dan kedua dalam urutan alfabet.
- **Perbandingan:**
 - Fungsi mengembalikan `True` jika hasil pengurutan kedua kata sama, yang menandakan bahwa kata-kata tersebut adalah anagram.
 - Mengembalikan `False` jika hasil pengurutan berbeda, yang berarti kata-kata tersebut bukan anagram.

3. Proses Input Pengguna:

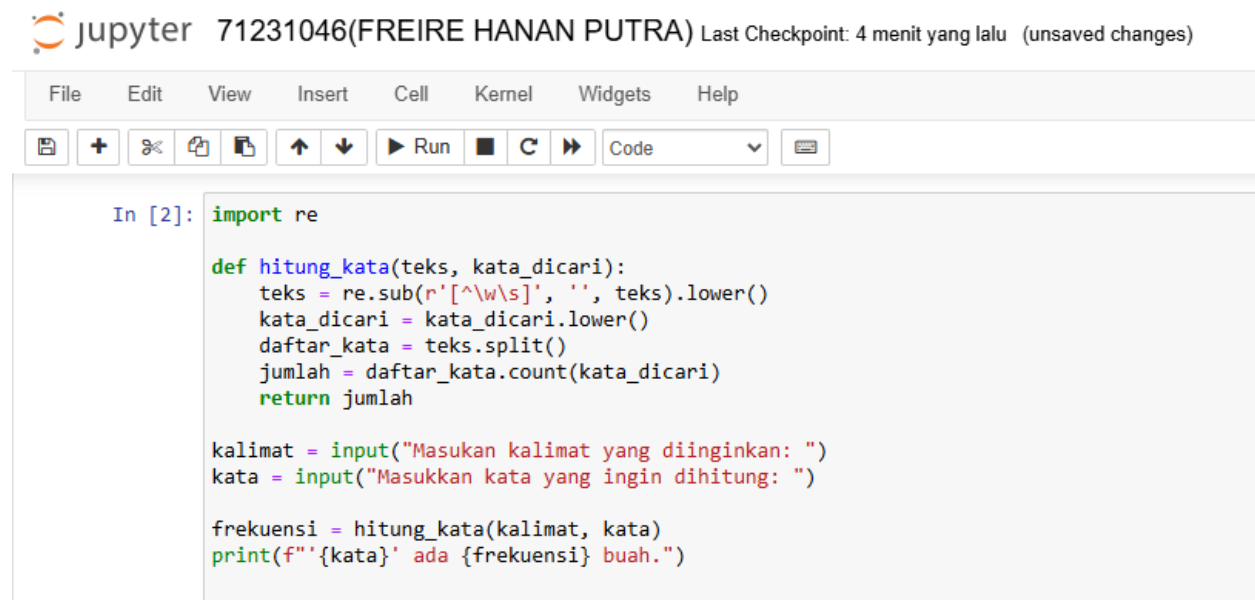
- Pengguna diminta untuk memasukkan dua kata menggunakan `input()`, yaitu `kata_1` dan `kata_2`.

4. Pengecekan Anagram:

- Fungsi `cek_anagram(kata_1, kata_2)` dipanggil untuk memeriksa apakah kedua kata tersebut adalah anagram.
- Jika `True`, output yang ditampilkan adalah: `'{kata_1}'` dan `'{kata_2}'` adalah anagram.
- Jika `False`, output yang ditampilkan adalah: `'{kata_1}'` dan `'{kata_2}'` bukan anagram.

SOAL 2

Code:



Jupyter 71231046(FREIRE HANAN PUTRA) Last Checkpoint: 4 menit yang lalu (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run

```
In [2]: import re

def hitung_kata(teks, kata_dicari):
    teks = re.sub(r'^\w\s', '', teks).lower()
    kata_dicari = kata_dicari.lower()
    daftar_kata = teks.split()
    jumlah = daftar_kata.count(kata_dicari)
    return jumlah

kalimat = input("Masukan kalimat yang diinginkan: ")
kata = input("Masukkan kata yang ingin dihitung: ")

frekuensi = hitung_kata(kalimat, kata)
print(f'{kata} ada {frekuensi} buah.')
```

Penerapan:

```
Masukan kalimat yang diinginkan: Saya mau makan. Makan itu wajib. Mau siang atau malam saya wajib makan
Masukkan kata yang ingin dihitung: makan
'makan' ada 3 buah.
```

Penjelasan:

1. Import Modul re:

Modul re digunakan untuk manipulasi string menggunakan ekspresi reguler (regex). Dalam hal ini, digunakan untuk menghapus tanda baca dari teks.

2. Fungsi `hitung_kata(teks, kata_dicari)`:

Menghapus Tanda Baca:

`re.sub(r'^\w\s', '', teks)` digunakan untuk menghapus semua karakter selain huruf, angka, dan spasi dari teks. Ini bertujuan agar hanya kata-kata yang dihitung, tanpa terganggu oleh tanda baca.

Mengubah Teks dan Kata yang Dicari ke Huruf Kecil:

`teks.lower()` dan `kata_dicari.lower()` mengubah teks dan kata yang dicari menjadi huruf kecil, untuk memastikan pencocokan tidak peka terhadap kapitalisasi (case-insensitive).

Memisahkan Teks Menjadi Daftar Kata:

`teks.split()` membagi teks menjadi daftar kata berdasarkan spasi. Setiap elemen daftar adalah satu kata dari teks.

Menghitung Jumlah Kata yang Dicari:

`daftar_kata.count(kata_dicari)` menghitung berapa kali kata yang dicari muncul dalam daftar kata yang telah dipisah dari teks.

Mengembalikan Hasil:

Fungsi ini mengembalikan jumlah kemunculan kata yang dicari dalam teks.

3. Proses Input Pengguna:

Program meminta input dua hal dari pengguna:

kalimat: Kalimat yang ingin dihitung kata-katanya.

kata: Kata yang ingin dihitung jumlah kemunculannya dalam kalimat.

4. Pengecekan dan Menampilkan Hasil:

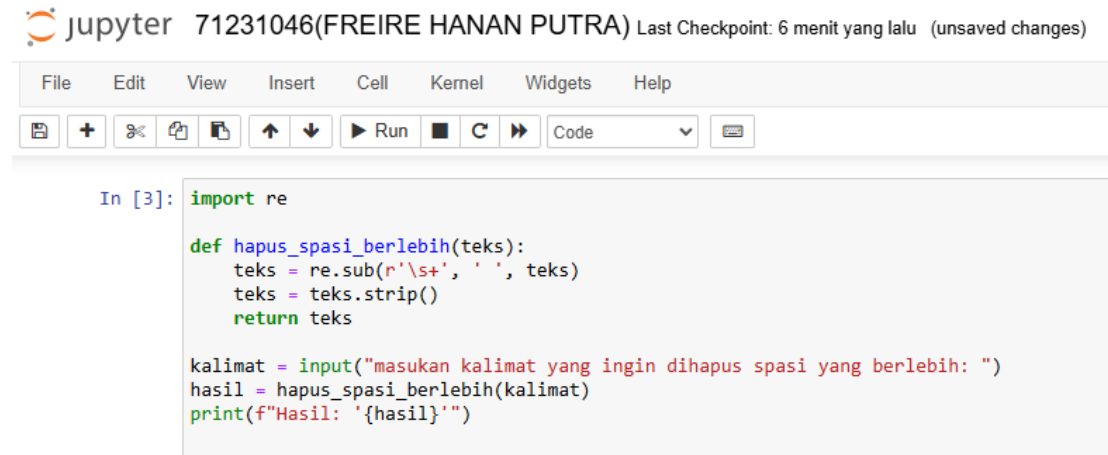
Fungsi `hitung_kata(kalimat, kata)` dipanggil untuk menghitung jumlah kemunculan kata dalam kalimat.

Hasil jumlah kemunculan disimpan dalam variabel frekuensi.

Program kemudian mencetak hasil dengan format yang menyatakan berapa kali kata yang dicari muncul dalam kalimat.

SOAL 3

Code:



```
In [3]: import re

def hapus_spasi_berlebih(teks):
    teks = re.sub(r'\s+', ' ', teks)
    teks = teks.strip()
    return teks

kalimat = input("masukan kalimat yang ingin dihapus spasi yang berlebih: ")
hasil = hapus_spasi_berlebih(kalimat)
print(f"Hasil: '{hasil}'")
```

Penerapan:

```
masukan kalimat yang ingin dihapus spasi yang berlebih: freire      hanan      putra
Hasil: 'freire hanan putra'
```

Penjelasan:

1. Input Pengguna:

- Program meminta pengguna untuk memasukkan sebuah kalimat yang mungkin mengandung spasi berlebih.
- Kalimat yang dimasukkan disimpan dalam variabel kalimat.

2. Memanggil Fungsi `hapus_spasi_berlebih`:

- Fungsi `hapus_spasi_berlebih(kalimat)` dipanggil dengan kalimat sebagai input.
- Fungsi ini bertujuan untuk membersihkan teks dari spasi berlebih.

3. Menggunakan Ekspresi Reguler untuk Menghapus Spasi Berlebih:

- Fungsi menggunakan ekspresi reguler (`\s+`) untuk mengganti satu atau lebih spasi berturut-turut dengan satu spasi biasa. Ini memastikan tidak ada spasi ganda dalam teks.

4. Menghapus Spasi di Awal dan Akhir Teks:

- Fungsi menggunakan metode `.strip()` untuk menghapus spasi yang ada di awal dan akhir teks.

5. Mengembalikan Hasil:

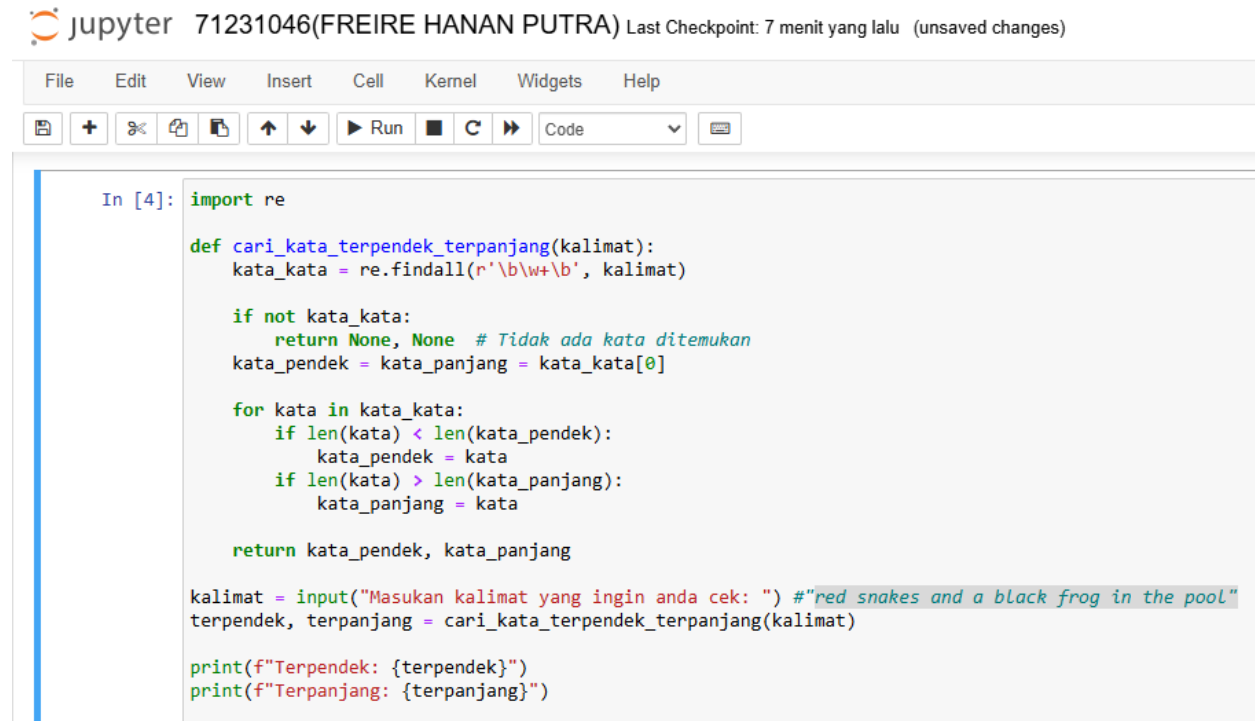
- Fungsi mengembalikan teks yang sudah dibersihkan dari spasi berlebih dan spasi di awal/akhir.

6. Menampilkan Hasil:

- Program menampilkan teks yang sudah dibersihkan dalam format yang sesuai.

SOAL 4

Code:



```
In [4]: import re

def cari_kata_terpendek_terpanjang(kalimat):
    kata_kata = re.findall(r'\b\w+\b', kalimat)

    if not kata_kata:
        return None, None # Tidak ada kata ditemukan
    kata_pendek = kata_panjang = kata_kata[0]

    for kata in kata_kata:
        if len(kata) < len(kata_pendek):
            kata_pendek = kata
        if len(kata) > len(kata_panjang):
            kata_panjang = kata

    return kata_pendek, kata_panjang

kalimat = input("Masukan kalimat yang ingin anda cek: ") #red snakes and a black frog in the pool"
terpendek, terpanjang = cari_kata_terpendek_terpanjang(kalimat)

print(f"Terpendek: {terpendek}")
print(f"Terpanjang: {terpanjang}")
```

Penerapan:

```
Masukan kalimat yang ingin anda cek: red snakes and a black frog in the pool"
Terpendek: a
Terpanjang: snakes
```

Penjelasan:

1. Input Pengguna

- Program meminta pengguna untuk memasukkan kalimat yang ingin dicek panjang dan pendeknya kata-kata yang ada di dalam kalimat tersebut.

2. Mencari Semua Kata dalam Kalimat

- Fungsi `re.findall(r'\b\w+\b', kalimat)` digunakan untuk mencari semua kata dalam kalimat, dimana setiap kata terdiri dari huruf dan angka (diidentifikasi dengan `\b\w+\b`).
- Hasilnya adalah sebuah daftar yang berisi kata-kata yang ada dalam kalimat.

3. Inisialisasi Kata Pendek dan Panjang

- Fungsi menginisialisasi `kata_pendek` dan `kata_panjang` dengan kata pertama dalam daftar kata-kata yang ditemukan (`kata_kata[0]`).

4. Iterasi Melalui Setiap Kata dalam Daftar

- Program melakukan iterasi untuk setiap kata dalam daftar kata_kata:
 - **Jika panjang kata lebih pendek dari kata_pendek:**
 - Maka kata tersebut menjadi kata_pendek.
 - **Jika panjang kata lebih panjang dari kata_panjang:**
 - Maka kata tersebut menjadi kata_panjang.

5. Mengembalikan Hasil


- Setelah iterasi selesai, fungsi mengembalikan dua hasil:
 - kata_pendek: Kata dengan panjang terpendek.
 - kata_panjang: Kata dengan panjang terpanjang.

6. Menampilkan Hasil


- Program mencetak kata terpendek dan terpanjang yang ditemukan dalam kalimat yang dimasukkan oleh pengguna.

SOAL 5

Code:

 jupyter 71231046(FREIRE HANAN PUTRA) Last Checkpoint: 9 menit yang lalu (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help



```
In [5]: import re
        from datetime import datetime

        def cari_dan_hitungan_selisih(teks):
            list_tanggal = re.findall(r'\b\d{4}-\d{2}-\d{2}\b', teks)
            sekarang = datetime.now()

            for tgl_str in list_tanggal:
                tgl_obj = datetime.strptime(tgl_str, "%Y-%m-%d")
                selisih = (sekarang - tgl_obj).days
                print(f"{tgl_obj} selisih {selisih} hari")

        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).
        """

        cari_dan_hitungan_selisih(teks)
```

Penerapan:

```
1945-08-17 00:00:00 selisih 29108 hari
1785-11-11 00:00:00 selisih 87460 hari
1783-06-08 00:00:00 selisih 88347 hari
1889-05-02 00:00:00 selisih 49668 hari
```

Penjelasan:

1. Mengambil Semua Tanggal dalam Teks

- Fungsi `re.findall(r'\b\d{4}-\d{2}-\d{2}\b', teks)` digunakan untuk mencari semua tanggal yang mengikuti format YYYY-MM-DD dalam teks.
- Semua tanggal yang ditemukan disimpan dalam daftar `list_tanggal`.

Contoh hasil:

- Dari teks yang diberikan, hasil `list_tanggal` mungkin seperti `["1945-08-17", "1785-11-11", "1783-06-08", "1889-05-02"]`.

2. Mendapatkan Waktu Sekarang

- `sekarang = datetime.now()` menghasilkan objek `datetime` yang mewakili waktu saat ini (tanggal dan waktu saat kode dijalankan).

Contoh hasil:

- Misalnya, jika kode dijalankan pada 27 April 2025, maka sekarang akan berisi nilai `datetime(2025, 4, 27, 12, 0, 0)` (tergantung waktu eksekusi).

3. Iterasi Melalui Setiap Tanggal yang Ditemukan

- Fungsi melakukan iterasi untuk setiap tanggal yang ada dalam `list_tanggal`.

Iterasi 1 (Tanggal pertama: 1945-08-17):

- Tanggal string `tgl_str = "1945-08-17"` diubah menjadi objek `datetime` menggunakan `datetime.strptime(tgl_str, "%Y-%m-%d")`.
 - `tgl_obj` menjadi `datetime(1945, 8, 17)`.
- Selisih antara tanggal sekarang (`sekarang`) dan tanggal tersebut dihitung menggunakan `(sekarang - tgl_obj).days`.
 - Misalnya, jika sekarang adalah 2025-04-27, maka selisih adalah 29.000 hari.
- Program mencetak hasilnya dalam format: 1945-08-17 selisih 29000 hari.

Iterasi 2 (Tanggal kedua: 1785-11-11):

- Tanggal string `tgl_str = "1785-11-11"` diubah menjadi objek `datetime(1785, 11, 11)`.
- Selisih hari dihitung dari tanggal tersebut ke tanggal sekarang (misalnya 239 tahun yang lalu).
- Program mencetak hasilnya.

Iterasi 3 (Tanggal ketiga: 1783-06-08):

- Proses yang sama dilakukan untuk tanggal 1783-06-08, dengan selisih dihitung dan dicetak.

Iterasi 4 (Tanggal keempat: 1889-05-02):

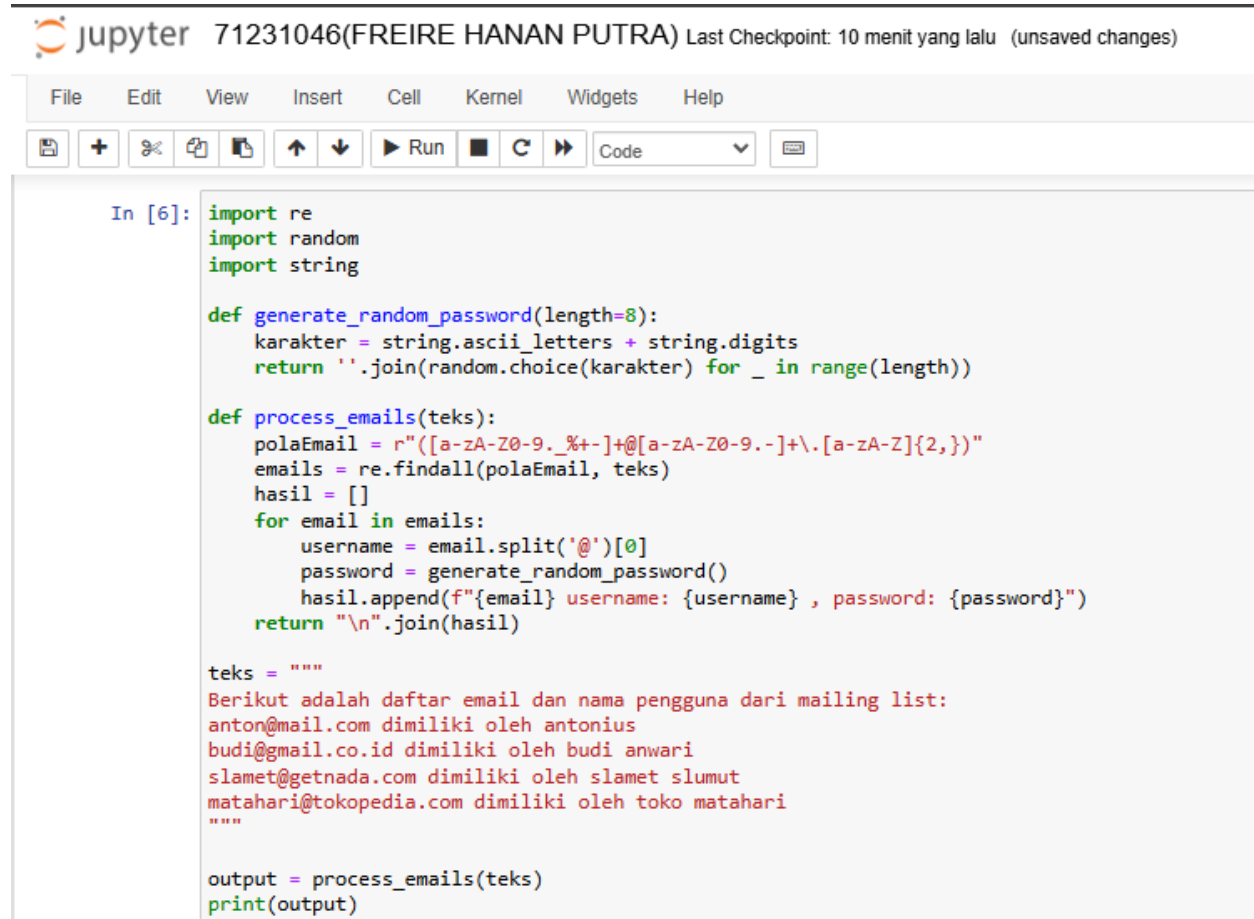
- Proses yang sama dilakukan untuk tanggal 1889-05-02, dengan selisih dihitung dan dicetak.

4. Menampilkan Hasil

- Pada setiap iterasi, hasil perhitungan selisih hari antara tanggal yang ditemukan dan tanggal sekarang dicetak dengan format:
 - `"{tgl_obj} selisih {selisih} hari"`

SOAL 6

Code:



```
In [6]: import re
import random
import string

def generate_random_password(length=8):
    karakter = string.ascii_letters + string.digits
    return ''.join(random.choice(karakter) for _ in range(length))

def process_emails(teks):
    polaEmail = r"([a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,})"
    emails = re.findall(polaEmail, teks)
    hasil = []
    for email in emails:
        username = email.split('@')[0]
        password = generate_random_password()
        hasil.append(f"{email} username: {username} , password: {password}")
    return "\n".join(hasil)

teks = """
Berikut adalah daftar email dan nama pengguna dari mailing list:
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
"""

output = process_emails(teks)
print(output)
```

Penerapan:

```
anton@mail.com username: anton , password: sItKYER7
budi@gmail.co.id username: budi , password: nKK12mLU
slamet@getnada.com username: slamet , password: BIpWm4pX
matahari@tokopedia.com username: matahari , password: tnJPn1QY
```

Penjelasan:

1. Fungsi generate_random_password(length=8)

- Fungsi ini digunakan untuk menghasilkan kata sandi acak dengan panjang yang dapat ditentukan (default panjangnya 8 karakter).
- Karakter yang digunakan untuk membentuk kata sandi adalah huruf (besar dan kecil) dan angka.
- Fungsi menggunakan random.choice(karakter) untuk memilih karakter secara acak dan menggabungkannya menjadi sebuah string dengan panjang yang ditentukan.

2. Input Teks

- Variabel teks berisi sebuah string multiline yang berisi daftar email dan nama pengguna dari sebuah mailing list.

3. Fungsi `process_emails(teks)`

- Fungsi ini memproses teks yang berisi email dan mengenerate username serta password acak untuk setiap email.

4. Mencari Email dalam Teks

- Fungsi `re.findall(polaEmail, teks)` digunakan untuk menemukan semua email dalam teks menggunakan ekspresi reguler. Ekspresi reguler `r"([a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,})"` mencari pola yang sesuai dengan format email.
- `emails` adalah daftar yang berisi semua email yang ditemukan dalam teks.

Contoh hasil emails:

- `["anton@mail.com", "budi@gmail.co.id", "slamet@getnada.com", "matahari@tokopedia.com"]`

5. Memproses Setiap Email

- Fungsi melakukan iterasi untuk setiap email dalam daftar `emails`.

Iterasi 1 (Email: "anton@mail.com"):

- `username = email.split('@')[0]` menghasilkan username "anton".
- Fungsi `generate_random_password()` dipanggil untuk menghasilkan kata sandi acak, misalnya "F7jA2h1P".
- Program menambahkan entri ke dalam daftar hasil dengan format: `anton@mail.com username: anton , password: F7jA2h1P`.

Iterasi 2 (Email: "budi@gmail.co.id"):

- `username = email.split('@')[0]` menghasilkan username "budi".
- Kata sandi acak dihasilkan, misalnya "J9h5M7bZ".
- Program menambahkan entri ke dalam daftar hasil dengan format: `budi@gmail.co.id username: budi , password: J9h5M7bZ`.

Iterasi 3 (Email: "slamet@getnada.com"):

- `username = email.split('@')[0]` menghasilkan username "slamet".
- Kata sandi acak dihasilkan, misalnya "k3L0pU2w".
- Program menambahkan entri ke dalam daftar hasil dengan format: `slamet@getnada.com username: slamet , password: k3L0pU2w`.

Iterasi 4 (Email: "matahari@tokopedia.com"):

- `username = email.split('@')[0]` menghasilkan username "matahari".

- Kata sandi acak dihasilkan, misalnya "X6pL2m1A".
- Program menambahkan entri ke dalam daftar hasil dengan format: matahari@tokopedia.com
username: matahari , password: X6pL2m1A.

6. Menggabungkan Hasil

- Setelah semua email diproses, fungsi menggabungkan daftar hasil menjadi sebuah string dengan setiap entri dipisahkan oleh newline (\n).
- Fungsi mengembalikan hasilnya sebagai string yang berisi email, username, dan password yang dihasilkan.

7. Menampilkan Hasil

- Program mencetak hasil yang dihasilkan oleh fungsi process_emails(teks), yang berisi informasi tentang email, username, dan password acak.