Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**<71230989>**
Nama Lengkap	**<Yohanes Nevan Adventus Wibawa>**
Minggu ke / Materi	**07 / Pengolahan String**

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%)

Pada bagian ini, tuliskan kembali semua materi yang telah anda pelajari minggu ini. Sesuaikan penjelasan anda dengan urutan materi yang telah diberikan di saat praktikum. Penjelasan anda harus dilengkapi dengan contoh, gambar/ilustrasi, contoh program (source code) dan outputnya. Idealnya sekitar 5-6 halaman.

Pengantar String

String adalah serangkaian karakter yang digabungkan menjadi satu kesatuan dan digunakan dalam program komputer untuk menyimpan teks, baik itu dalam bentuk pendek maupun panjang. Dalam representasi internal, string adalah kumpulan karakter yang disimpan dalam kode ASCII. Meskipun tidak semua bahasa pemrograman memiliki tipe data string, seperti bahasa C misalnya, namun tipe data ini adalah jenis data yang penting karena mampu menyimpan huruf atau karakter. Secara konseptual, string bukanlah tipe data dasar karena dapat menyimpan lebih dari satu nilai tunggal sebagai satu kesatuan. Beberapa bahasa pemrograman menggambarkan string sebagai kumpulan karakter atau array karakter.

Pengaksesan String dan Manipulasi String

```
namasaya = "Antonius Rachmat C"
temansaya1 = "Yuan Lukito"
temansaya2 = 'Laurentius Kuncoro'
temansaya3 = "Matahari" + 'Bakti'
temansaya4 = "Yohanes" + 'Nevan' + 'AW'

print(temansaya4)
print(temansaya3)
print(namasaya[0]) #'A'
print(namasaya[9]) #'R'
print(temansaya1[1]) #'u'
huruf = temansaya2[0]
print(huruf) #'L'
```

```
PS C:\Users\LENOVO\Tugas PrakAlPro\Tu
O/Tugas PrakAlPro/Tugas 7/coba.py"
YohanesNevanAW
MatahariBakti
A
R
u
L
```

String pertama kali dibuat dengan mendeklarasikan variabel dan kemudian mengisinya dengan data teks. Setelah itu, string dapat diakses sebagai satu kesatuan dengan menyebut nama variabelnya, atau dapat diakses per huruf dengan menyebutkan indeksnya. Indeks pada string dimulai dari 0, mirip dengan indeks pada list. Penting untuk dicatat bahwa indeks string haruslah berupa bilangan bulat, dan tidak bisa pecahan. Dalam memori komputer, string disimpan secara berurutan menggunakan struktur data yang mirip dengan list, dimana setiap huruf disimpan dengan indeks yang dimulai dari nol.

Operator dan Metode String

Operator in

Pada String kita dapat memeriksa apakah suatu kalimat merupakan substring dari suatu kalimat

lain dengan menggunakan operator in. Hasil dari operator ini adalah True / False.

```
kalimat = "saya mau makan"
data = "saya"
print(data in kalimat) #True
print("mau" in kalimat) #True
print("dia" in kalimat) #False
```

```
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 7>
O/Tugas PrakAlPro/Tugas 7/coba.py"
True
True
False
```

Selain operator in, pada String juga dapat dilakukan perbandingan (comparison) yang juga menghasilkan True atau False.

```
if "saya" > "dia":
    print("Ya") #Ya
else:
    print("Tidak")

if "dua" == "dua":
    print("Sama") #Sama
```

Pertama-tama, mari kita bahas setiap bagian dari kode tersebut:

Pada bagian pertama, terdapat perbandingan antara dua string: "saya" dan "dia". Kita dapat melihat bahwa "saya" secara leksikal (berdasarkan urutan abjad) berada setelah "dia". Oleh karena itu, ekspresi ini menghasilkan nilai True.

Pada bagian kedua, terdapat perbandingan antara dua string: "dua" dan "dua". Karena kedua string tersebut identik dan memiliki nilai yang sama, ekspresi ini menghasilkan nilai True.

Jadi, penjelasannya adalah:

Pada bagian pertama, karena "saya" secara leksikal berada setelah "dia", maka ekspresi tersebut menghasilkan nilai True, sehingga outputnya adalah "Ya".
Pada bagian kedua, karena kedua string "dua" dan "dua" identik, maka ekspresi tersebut menghasilkan nilai True, sehingga outputnya adalah "Sama".

Fungsi len

Untuk mengetahui jumlah karakter dalam sebuah string, digunakan operator len(<string>). Untuk menampilkan huruf terakhir dari string, kita perlu menggunakan indeks string yang merupakan panjang string dikurangi satu (len(<string>) - 1), karena indeks dimulai dari 0. Berikut adalah contoh program Python yang mengilustrasikan hal ini:

```
kalimat = "universitas kristen duta wacana yogyakarta"
print(len(kalimat)) #output 42

terakhir = kalimat[len(kalimat)-1]
print(terakhir) #output 'a'

#bisa juga menggunakan indeks -1
terakhir_versi2 = kalimat[-1]
print(terakhir_versi2) #output 'a'
#atau menggunakan indeks -2 untuk huruf terakhir kedua
terakhir2 = kalimat[-2]
print(terakhir2) #output 't'
```

```
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas
O/Tugas PrakAlPro/Tugas 7/coba.py"
42
a
a
t
```

Traversing String

String slice adalah menampilkan substring pada sebuah string dengan menggunakan indeks dari awal tertentu sampai akhir-1 tertentu. Sintaksnya menggunakan <string>[awal:akhir]. Bagian awal atau akhir boleh dikosongkan. Bagian awal dimulai dari 0.

```
kalimat = "indonesia jaya"
i = 0
while i < len(kalimat):
    print(kalimat[i],end='')
i += 1
```

String Slice

String slice adalah teknik untuk menampilkan potongan dari sebuah string dengan menggunakan indeks yang menentukan awal dan akhir dari potongan tersebut. Sintaksisnya adalah <string>[awal:akhir]. Bagian awal atau akhir dapat diabaikan. Indeks awal dimulai dari 0.

```
kalimat = "cerita rakyat"
awal = 0
akhir = 6
print(kalimat[awal:akhir]) #cerita
print(kalimat[7:len(kalimat)]) #rakyat
print(kalimat[:5]) #cerit
print(kalimat[5:]) #a rakyat
print(kalimat[:]) #cerita rakyat
```

```
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 7> & "
O/Tugas PrakAlPro/Tugas 7/coba.py"
cerita
rakyat
cerit
a rakyat
cerita rakyat
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 7>
```

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 dinputkan	string.endswith()
startswith()	mengetahui apakah suatu string diawali dengan string yang dinputkan	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()

Operator * dan + pada string

Pada Python operator + dan * memiliki kemampuan khusus. Operator + yang biasanya digunakan untuk menjumlahkan bilangan bisa digunakan untuk menggabungkan dua buah string. Sedangkan operator * yang bisa digunakan untuk mengkalikan bilangan bisa digunakan untuk menampilkan string sejumlah perkaliannya. Perhatikan kode berikut:

```
kata1 = "saya"
kata2 = "makan"
kata3 = kata1 + " " + kata2
print(kata3) #hasil adalah penggabungan: saya makan
kata4 = "ulang"
print(kata4 * 4) #hasil adalah ulangulangulangulang
kata4 = "ulang "
print(kata4 * 2) #hasil adalah ulang ulang
```

```
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 7> &
O/Tugas PrakAlPro/Tugas 7/coba.py"
saya makan
ulangulangulangulang
ulang ulang
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 7>
```

Parsing String

Parsing string adalah proses menelusuri sebuah string secara berurutan untuk mendapatkan, menemukan, atau mengubah bagian-bagian tertentu dari string tersebut. Misalnya, dalam kalimat "Saudara-saudara, pada tanggal 17-08-1945 Indonesia merdeka", kita ingin mengambil tanggal bulan tahun dan mengubahnya menjadi format 08/17/1945. Salah satu cara untuk melakukan parsing string adalah sebagai berikut: Memisahkan string menjadi token-token dengan menggunakan spasi sebagai pemisah, sehingga kita memperoleh: "Saudara-saudara", "pada", "tanggal", "17-08-1945", "Indonesia", dan "merdeka".
Melanjutkan dengan mencari token yang diawali dengan angka, kemudian memisahkan angka tersebut dengan pemisah '-'.
Mengatur ulang token-token yang telah ditemukan sehingga membentuk format yang diinginkan.
Ini adalah cara yang dapat digunakan untuk menyelesaikan tugas parsing string seperti yang dijelaskan di atas, contohnya dibawah ini.

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

```
O/Tugas PrakAlPro/Tugas 7/coba.py"
08/17/1945
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 7>
```

Kegitan Praktikum 7.1

Untuk menghitung huruf hidup digunakan tahapan sebagai berikut:

• minta input kalimat yang akan dicari
• ubah string menjadi huruf kecil semua agar tidak membedakan huruf besar dan kecil
• siapkan variabel total jumlah huruf hidup aiueo dan set nilai awal = 0
• carilah ada berapa huruf 'a','i','u','e','o' pada kalimat yang diinputkan dengan method count()

Hasil:

```
a_string = "AnTonIus"
lowercase = a_string.lower()
total = 0
for x in "aiueo":
    jml = lowercase.count(x)
    total += jml
print(total) #hasil = 4
```

```
PS C:\Users\LENOVO\Tugas P
O/Tugas PrakAlPro/Tugas 7/
4
PS C:\Users\LENOVO\Tugas P
```

Kegiatan Praktikum 7.2

Kode tersebut berfungsi untuk mengubah format tanggal dari "YYYY-MM-DD" menjadi "DD-MM-YYYY". Prosesnya adalah sebagai berikut:

1. Tanggal "2020-12-01" dipisahkan menggunakan metode split() dengan pemisah "-" sehingga menghasilkan list ["2020", "12", "01"] yang disimpan dalam variabel hasil.
2. Kemudian, tgl2 dibentuk dengan mengambil elemen terakhir ("01") dari hasil (hasil[2]), diikuti dengan "-" sebagai pemisah, diikuti dengan elemen kedua ("12") dari hasil (hasil[1]), diikuti lagi dengan "-", dan terakhir elemen pertama ("2020") dari hasil (hasil[0]).
3. Hasil akhir, yaitu tgl2, dicetak.

Jadi, kode tersebut mengubah format tanggal dari "YYYY-MM-DD" menjadi "DD-MM-YYYY".

```
tgl = "2020-12-01"
hasil = tgl.split("-")
tgl2 = hasil[2]+"-"+hasil[1]+"-"+hasil[0]
print(tgl2)
```

```
O/Tugas PrakAlPro/Tugas 7/
01-12-2020
PS C:\Users\LENOVO\Tugas P
```

Kegiatan Praktikum 7.3

Kode ini bertujuan untuk mengecek apakah sebuah string merupakan palindrom atau tidak.

Langkah-langkahnya adalah sebagai berikut:

1. String satu diubah menjadi huruf kecil menggunakan metode `lower()` untuk menyamakan huruf besar-kecil.
2. Menggunakan list comprehension, string satu difilter sehingga hanya karakter alfabet yang tetap, dan kemudian dijadikan satu kembali menggunakan metode join().
3. Variabel dua dibuat dengan mengambil string satu yang telah difilter dan dibalik menggunakan slicing dengan langkah -1 (satu[::-1]).
4. Dilakukan pengecekan apakah dua sama dengan satu. Jika sama, maka string tersebut merupakan palindrom, dan jika tidak, bukan.

Jadi, kode ini memeriksa apakah string setelah difilter dan dibalik masih sama dengan string aslinya.

```
satu = "Step! on, no.. pets??"
satu = satu.lower()

satu = ''.join([i for i in satu if i.isalpha()]) #buang semua yang bukan alfabet

dua = satu[::-1]
if dua == satu:
    print("palindrom")
else:
    print("bukan")
```

```
O/Tugas PrakAlPro/Tugas 7/
palindrom
PS C:\Users\LENOVO\Tugas P
```

Kegiatan Praktikum 7.4

Fungsi `ambil_kata_kalimat` ini bertujuan untuk mengambil setiap rangkaian kata sepanjang `n` dari sebuah kalimat dan mengembalikannya dalam bentuk list. Berikut adalah penjelasan singkatnya:

1. Kalimat awal diubah menjadi huruf kecil menggunakan metode `lower()` agar tidak ada perbedaan antara huruf besar dan huruf kecil.
2. Variabel `hasil_akhir` didefinisikan sebagai list kosong untuk menampung hasil akhir.
3. Kalimat dipecah menggunakan metode `split()` untuk memisahkan kata-kata berdasarkan spasi.
4. Dilakukan iterasi menggunakan loop `for` untuk mengambil setiap kata sepanjang `n`.
5. Dalam setiap iterasi, dilakukan penggabungan kembali (`join()`) untuk mendapatkan rangkaian kata sepanjang `n`, dan hasilnya dimasukkan ke dalam list `hasil_akhir`.
6. Setelah iterasi selesai, list `hasil_akhir` dikembalikan sebagai output dari fungsi.

Jadi, jika dipanggil dengan argumen `ambil_kata_kalimat(text, 2)`, fungsi ini akan mengembalikan setiap rangkaian dua kata dari kalimat yang telah diubah menjadi huruf kecil.

```
text = 'A quick brown fox jumps over the lazy dog.'

def ambil_kata_kalimat(kalimat, n):
    kalimat = kalimat.lower() #ubah huruf kecil
    hasil_akhir = [] #siapkan tempat hasil
    hasil = kalimat.split() #pecah berdasarkan spasi
    for i in range (0,len(hasil)): #loop per hasil pecah
        tmp = ' '.join(hasil[i:i + n]) #ambil per indeks
        hasil_akhir.append(tmp) #tambahkan ke list

    return hasil_akhir #return

print(ambil_kata_kalimat(text, 2))
```

```
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 7> & "C:/Program Files/Python312/python.exe" "c:/Users/LENOV
O/Tugas PrakAlPro/Tugas 7/kegiatan praktikum 7.4.py"
['a quick', 'quick brown', 'brown fox', 'fox jumps', 'jumps over', 'over the', 'the lazy', 'lazy dog.
', 'dog.']
PS C:\Users\LENOVO\Tugas PrakAlPro\Tugas 7>
```

BAGIAN 2: LATIHAN MANDIRI (60%)

Pada bagian ini anda menuliskan jawaban dari soal-soal Latihan Mandiri yang ada di modul praktikum. Jawaban anda harus disertai dengan source code, penjelasan dan screenshot output.

SOAL 7.1

```
def cek_anagrams():
    kata1 = input("Masukkan kata pertama: ")
    kata2 = input("Masukkan kata kedua: ")

    kata1 = kata1.replace(" ", "").lower()
    kata2 = kata2.replace(" ", "").lower()

    kata1_sorted = sorted(kata1)
    kata2_sorted = sorted(kata2)

    if kata1_sorted == kata2_sorted:
        print(f"'{kata1}' dan '{kata2}' adalah anagram.")
    else:
        print(f"'{kata1}' dan '{kata2}' bukan anagram.")

cek_anagrams()
```

Output :

```
'atma' dan 'maat' adalah anagram.
```

Penjelasan :

Program Python ini bertugas memeriksa apakah dua kata yang dimasukkan pengguna adalah anagram satu sama lain. Fungsi tunggal `check_anagrams()` meminta pengguna memasukkan dua kata, kemudian menghilangkan spasi dan mengubah huruf menjadi huruf kecil untuk mempermudah perbandingan. Selanjutnya, huruf-huruf dalam kedua kata diurutkan. Jika urutan hurufnya sama, kata-kata tersebut dianggap anagram dan program akan mencetak pesan yang menyatakan bahwa kata-kata tersebut adalah anagram. Jika tidak, program akan mencetak pesan bahwa kata-kata tersebut bukan anagram. Program diakhiri dengan pemanggilan fungsi `check_anagrams()`, sehingga pengguna diminta memasukkan dua kata dan program menentukan apakah keduanya adalah anagram atau tidak, serta mencetak hasilnya ke layar.

Soal 7.2

```python
def hitung_kemunculan_kata(kalimat, kata):

    kalimat = kalimat.lower()
    kata = kata.lower()

    count = kalimat.count(kata)
    return count

kalimat = "Saya mau makan. Makan itu wajib. Mau siang atau malam saya wajib
makan"
kata_dicari = "makan"

jumlah_kemunculan = hitung_kemunculan_kata(kalimat, kata_dicari)
print(f"{kata_dicari} ada {jumlah_kemunculan} buah")
```

Output :

```
✓ 0.0s
makan ada 3 buah
```

Penjelasan:

Program ini menggunakan metode `count()` pada string untuk menghitung berapa kali suatu kata muncul dalam sebuah kalimat. Pertama, program mengonversi kalimat dan kata pencarian menjadi huruf kecil untuk memperhitungkan kemungkinan perbedaan kapitalisasi. Kemudian, program menghitung frekuensi kemunculan kata tersebut dalam kalimat menggunakan metode `count()`. Hasilnya dicetak ke layar.

Soal 7.3

```
def hapus_spasi_berlebih(kalimat):

    kata_kalimat = kalimat.split()

    kalimat_baru = ' '.join(kata_kalimat)

    return kalimat_baru

kalimat = "saya tidak suka   memancing ikan "
kalimat_tanpa_spasi_berlebih = hapus_spasi_berlebih(kalimat)
print(kalimat_tanpa_spasi_berlebih)
```

Output:

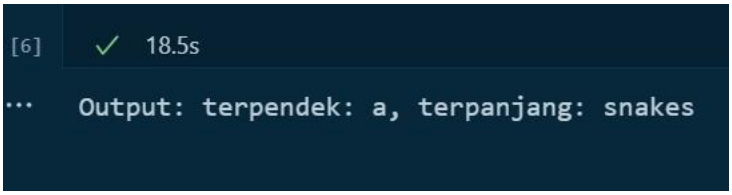
Penjelasan:

Fungsi hapus_spasi_berlebih(kalimat) menerima sebuah string sebagai input.
String tersebut dipecah menjadi kata-kata menggunakan metode split() untuk menghilangkan spasi berlebih.
Kemudian, kata-kata yang telah dipisahkan digabungkan kembali menjadi satu string menggunakan metode join() dengan satu spasi di antara kata-kata.
Hasilnya adalah string tanpa spasi berlebih, yang kemudian dikembalikan sebagai output.
Dengan demikian, program akan menghasilkan string yang sudah diubah dengan menghapus semua spasi berlebih menjadi satu spasi normal di antara kata-kata.

Soal 7.4

```python
def terpendek_terpanjang_kata(sentence):
    words = sentence.split()

    terpendek_kata = min(words, key=len)
    terpanjang_kata = max(words, key=len)

    return terpendek_kata, terpanjang_kata

kalimat = input("Masukkan kalimat: ")

terpendek, terpanjang = terpendek_terpanjang_kata(kalimat)

print("Output: terpendek:", terpendek + ",", "terpanjang:", terpanjang)
```

Output:



Penjelasan:

Program ini bertujuan untuk menemukan kata terpendek dan terpanjang dari sebuah kalimat yang dimasukkan oleh pengguna.

Fungsi ` terpendek_terpanjang_kata(sentence)'menerima sebuah kalimat sebagai argumen. Pertama, kalimat tersebut dipisahkan menjadi kata-kata menggunakan metode `split()`. Selanjutnya, kita menggunakan fungsi `min()` dan `max()` dengan argumen `key=len` untuk mencari kata terpendek dan terpanjang dalam list kata-kata tersebut. Fungsi `min()` akan mengembalikan kata terpendek berdasarkan panjangnya, sedangkan `max()` akan mengembalikan kata terpanjang.

Setelah mendapatkan kata terpendek dan terpanjang, program mencetak hasilnya ke layar.

Pada bagian utama program, pengguna diminta untuk memasukkan kalimat. Kemudian, fungsi ` terpendek_terpanjang_kata(sentence)' dipanggil dengan kalimat yang dimasukkan pengguna sebagai argumen. Hasilnya, kata terpendek dan terpanjang, dicetak ke layar.

Dengan demikian, program ini memberikan informasi tentang kata terpendek dan terpanjang dalam kalimat yang dimasukkan pengguna.

Link Github:

Link = https://github.com/YohanesNevan/Tugas-Laporan-AlPro-7.git