Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71230985**
Nama Lengkap	**TOMAS BECKET**
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%)

MATERI 1

String adalah sebuah karakter yang menjadi kesatuan dan digunakan dalam program komputer untuk menyimpan kalimat, baik itu panjang ataupun pendek sebuah kalimat tersebut. Dan biasanya string adalah jenis data yang mampu menyimpan huruf atau karakter yang dapat disimpan dlam kode ASCII dan tidak semua bahasa pemgrograman memiliki tipe data string, contohnya yaitu baha C

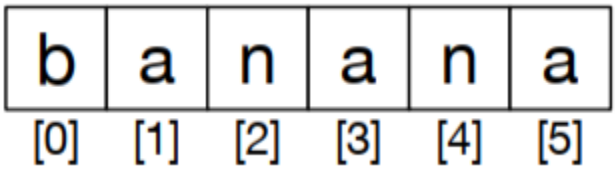
PENGAKSESAN STRING DAN MANIPULASI STRING

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

print(temansaya3)
print(namasaya[0]) #'A'
print(namasaya[9]) #'R'
print(temansaya1[1]) #'u'

huruf = temansaya2[0]
print(huruf) #'L'
```

Pada source code pada buku modul menunjukan bahwa kode tersebut ingin memprint huruf yang terdapat pada keempat variabel dimana saat print(namasaya[0]) maka akan muncul huruf A yaitu index 0 dari namasaya [A]ntonius Rachmat C. lalu jika dibuat variabel baru untuk digunakan pada fungsi print namun pada variabel baru tersebut berisi sebuah index yang dipilih maka tidak mengubah pilihan atau hasil output yang berbeda.



Berikut bentuk string yang di simpan dalam memory, index pada string dimulai dari 0 layaknya sebuah list. Dan indeks string haruslah berupa sebuah bilangan bulat, bukan pecahan.

MATERI 2

Pada string juga dapat memeriksa suatu kaliat merupakan substring atau bukan dan hasil operator hanya menunjukan True/False.

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

Misal pada variabel kalimat tersimpan string “saya mau makan” dan print “mau” in kalimat maka akan menghasilkan nilai True karena terdapat kata mau pada variabel kalimat sedangkan ketika print “dia” in kalimat akan muncul False karena tidak terdapat huruf dia pada variabel Kalimat.

```
# Operator IN
# hanya memprint True atau False saja
if "saya" > "dia":
    print ("Ya")
else:
    print("Tidak")

if "dua" == "tiga":
    print("sama")
```

Contoh lain penggunaan percabangan menggunakan string. String pada percabangan tidak terlalu efektif kecuali menggunakan sebuah nilai atau integer.

MATERI 3

Fungsi len digunakan untuk mengetahui berapa panjang (berapa jumlah karakter) dari sebuah string dengan menggunakan operator len(<string>). untuk menampilkan huruf terakhir dari sebuah string bisa menggunakan len(<string>-1)

```
# Fungsi Len
kalimat = "universitas kristen duta wacana yogyakarta"
print(len(kalimat))

terakhir = kalimat[len(kalimat)-1]
print(terakhir)
```

Ketika variabel kalimat dilakukan print(len(kalimat)) maka akan muncul sebuah angka yaitu 42 yang dimana angka tersebut menghitung total huruf dan spasi pada string di variabel kalimat.

Namun jika dibuat variabel baru terakhir dimana kalimat pada len(kalimat)-1 akan memprint huruf terakhir pada string kalimat yaitu a pada kata yogyakarta.

MATERI 4

TRAVERSING STRING

Untuk menampilkan string dengan cara ditampilkan huruf demi huruf adalah dengan menggunakan loop yang dilakukan per huruf dengan 2 cara yaitu dilakukan dengan indeks dan tanpa akses terhadap indeks secara otomatis.

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

print()

for kal in kalimat:
    print(kal, end="")
```

Source code diatas menjelaskan bahwa string pada variabel kalimat setiap hurufnya akan diprint satu persatu secara urut mulai dari index 0 hingga index terakhir yaitu total huruf dan spasi terhitung semua menggunakan looping atau perulangan dengan penambahan end=”” untuk memberi kejelasan huruf yang muncul.

MATERI 5

```
# String Slice
kalimat = "cerita rakyat"
awal = 0
akhir = 0
print(kalimat[awal:akhir])
print(kalimat[7:len(kalimat)])
print(kalimat[:5])
print(kalimat[5:])
print(kalimat[:])

# String bersifat immutable sehingga tidak bisa diubah dan hanya diinisialisasi
```

String slice adalah operasi yang digunakan untuk menampilkan bagian substring dari sebuah string dengan menggunakan indeks awal tertentu hingga indeks sebelum akhir tertentu. Sintaksisnya adalah <string>[awal:akhir]. Bagian awal atau akhir dapat dikosongkan, dimana bagian awal dimulai dari indeks 0.

Sehingga ketika melakukan print(kalimat[awal:akhhir]) maka index yang digunakan yaitu mulai dari index 0 sampai index 6(contoh source code diatas typo seharusnya akhir = 6) akan memunculkan kata “cerita”.

Sedangkan jika print(kalimat[:5]) maka yang muncul yaitu index 0 hingga index 5 = “cerit”

Satu hal penting tentang string bahwa string merupakan data yang bersifat immutable yang dimana data tersebut tidak dapat diubah selagi program itu berjalan dan hanya bisa diinisialisasi saja.

Sehingga jika ingin mengubah kalimat tersebut harus menggunakan variabel baru atau yang berbeda

```
kalimat = "satu"
kalimat_baru = kalimat[0] + "alah"   #salah
```

Ada pun beberapa method string yang sering digunakan contohnya: count, isdigit, split, islower, capitalize, dsbg.

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()

MATERI 6

Dalam Python, operator + memiliki kemampuan khusus yang memungkinkannya untuk menggabungkan dua buah string, bukan hanya menjumlahkan bilangan seperti pada operasi aritmatika biasa. Sedangkan operator * memiliki kemampuan khusus yang memungkinkannya untuk menampilkan string sejumlah perkaliannya, bukan hanya mengalikan bilangan seperti pada operasi aritmatika biasa.

```
# Operator * dan + pada String
kata1 = "saya"
kata2 = "makan"
kata3 = kata1 + " " + kata2
print(kata3)
kata4 = "ulang"
print(kata4 * 4)
kata4 = "ulang "
print(kata4 * 2)
```

Ketika terdapat 3 variabel dimana kata1 = saya dan kata2 = makan dan kata3 = kata1 + “ “ + kata2 maka akan menghasilkan penggabungan kata saya dan makan menjadi “saya makan” penggunaan “ “ pada tengah-tengah + menjadi spasi pada sebuah string karena jika tidak digunakan maka kedua kata tersebut tidak memiliki spasi, sedangkan ketika kata dikali dengan sebuah integer atau bilangan angka maka akan terjadi pengulangan kata seperti fungsi pada line 6 akan memunculkan kata ulang sebanyak 4x tanpa spasi.

MATERI 7

Parsing string adalah cara menelusuri string bagian demi bagian untuk mendapatkan/menemukan bagian string yang diinginkan contoh source codenya seperti berikut:

```
# Pharsing String
# menelusuri string bagian perbagian untuk menemukan atau mengubah bagian string
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])
```

BAGIAN 2: LATIHAN MANDIRI (60%)

SOAL 1

1.1 SOURCE CODE

```
# LATIHAN 1
kata1 = input("Masukkan kata pertama: ")
kata2 = input("Masukkan kata kedua: ")

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

sorted_kata1 = sorted(kata1)
sorted_kata2 = sorted(kata2)

if sorted_kata1 == sorted_kata2:
    print(f"{kata1} dan {kata2} adalah anagram.")
else:
    print(f"{kata1} dan {kata2} bukan anagram.")
```

1.2 PENJELASAN

Source code tersebut akan meminta input dua kata dari user, untuk kemudian menentukan apakah kata-kata tersebut merupakan anagram satu sama lain atau tidak.

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

Awalnya, user diminta memasukkan dua kata melalui fungsi input(). Setelah kata-kata tersebut dimasukkan, skrip ini kemudian menstandardisasi kedua kata ke bentuk huruf kecil menggunakan metode lower() dan mengeliminasi spasi dengan metode replace(" ", "").

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

Berikutnya, dengan menggunakan fungsi sorted(), skrip mengurutkan huruf-huruf dalam kedua kata tersebut, sehingga menghasilkan urutan karakter yang terorganisir alfabetis.

```
sorted_kata1 = sorted(kata1)
sorted_kata2 = sorted(kata2)
```

Pada langkah terakhir, skrip membandingkan kedua rangkaian karakter yang sudah diurutkan tersebut. Jika kedua rangkaian karakter identik, maka dianggap bahwa kedua kata tersebut adalah anagram dan skrip akan mencetak pesan konfirmasi. Sebaliknya, jika kedua rangkaian karakter tersebut berbeda, skrip akan mencetak pesan yang menyatakan bahwa kata-kata tersebut bukanlah anagram.

```
if sorted_kata1 == sorted_kata2:
    print(f"{kata1} dan {kata2} adalah anagram.")
else:
    print(f"{kata1} dan {kata2} bukan anagram.")
```

1.3 HASIL OUTPUT

```
Kata pertama: mata
Kata kedua: aamt
mata dan aamt adalah anagram.
```

```
Kata pertama: makan
Kata kedua: aknma
makan dan aknma adalah anagram.
```

```
Kata pertama: makan
Kata kedua: minum
makan dan minum bukan anagram.
```

SOAL 2

2.1 SOURCE CODE

```
# LATIHAN 2
text = input("Kalimat: ")
print()
target_word = input("Yang dicari: ")

text_lower = text.lower()
target_word_lower = target_word.lower()
word_count = text_lower.count(target_word_lower)

print()
print(f"{target_word} ada {word_count} buah")
```

2.2 PENJELASAN

Pada soal diminta untuk membuat suati program yang dapat menghitung frekuensi kemunculan suatu kata tertentu dalam sebuah kalimat yang dimasukkan oleh user.

```
text = input("Kalimat: ")
```

Dimana awlanya user akan diminta untuk memasukkan sebuah kalimat melalui fungsi input(), dengan pesan prompt "Kalimat: ". Kalimat yang dimasukkan kemudian disimpan dalam variabel text.

Setelah itu, memberikan baris kosong menggunakan print() tanpa argumen. Ini dilakukan untuk memisahkan input dari pengguna dengan informasi selanjutnya yang akan ditampilkan.

```
target_word = input("Yang dicari: ")
```

Kemudian User akan memasukkan kata yang ingin dicari frekuensinya dalam kalimat yang sudah dimasukkan sebelumnya. Kata yang dicari dimasukkan melalui fungsi input() dengan pesan prompt "Yang dicari: ", dan disimpan dalam variabel target_word.

```
text_lower = text.lower()
target_word_lower = target_word.lower()
```

Kode kemudian mengubah kedua input (kalimat dan kata yang dicari) menjadi huruf kecil menggunakan metode lower(). Hal ini dilakukan agar pencarian kata tidak dipengaruhi oleh perbedaan kapitalisasi. Hasil dari pengolahan ini disimpan dalam variabel text_lower dan target_word_lower.

```
word_count = text_lower.count(target_word_lower)
```

Untuk menemukan frekuensi kemunculan kata yang dicari dalam kalimat, kode menggunakan metode count() pada string text_lower, dengan target_word_lower sebagai argumennya. Metode count() ini mengembalikan jumlah berapa kali target_word_lower muncul dalam text_lower. Hasil perhitungan ini disimpan dalam variabel word_count.

```
print(f"{target_word} ada {word_count} buah")
```

Terakhir, program mencetak hasilnya dalam format yang telah ditentukan, yaitu menampilkan kata yang dicari beserta jumlah kemunculannya dalam kalimat. Format outputnya adalah "<kata yang dicari> ada <jumlah> buah".

2.3 HASIL OUTPUT

```
Kalimat: Saya mau makan. Makan itu wajib. Mau siang atau malam saya wajib makan

Yang dicari: makan

makan ada 3 buah
```

```
Kalimat: Jika saya makan, maka saya kenyang. jika saya tidak makan, makan saya tidak kenyang

Yang dicari: kenyang

kenyang ada 2 buah
```

SOAL 3

3.1 SOURCE CODE

```
# LATIHAN 3
string_input = input("Kalimat: ")

string_output = ' '.join(string_input.split())

print(string_output)
```

3.2 PENJELASAN

kode diatas dibuat mengambil kalimat dengan spasi berlebih, memprosesnya untuk menghilangkan spasi berlebih tersebut, dan menghasilkan versi kalimat yang lebih rapi dengan hanya satu spasi normal antara setiap kata.

```
string_output = ' '.join(string_input.split())
```

' '.join(string_input.split()) digunakan untuk menggabungkan kembali kata-kata dalam daftar tersebut menjadi sebuah string baru, dengan menggunakan spasi tunggal (' ') sebagai pemisah antar kata. Ini berarti, tidak peduli berapa banyak spasi berlebih yang ada di antara kata-kata dalam input asli, output akan selalu memiliki satu spasi normal di antara kata-kata.

3.3 HASIL OUTPUT

```
Kalimat: saya    tidak  suka  memancing  ikan
saya tidak suka memancing ikan
```

SOAL 4

4.1 SOURCE CODE

```
# LATIHAN 4
kalimat = input("Kalimat: ")

kata = kalimat.split()
terpendek = terpanjang = kata[0]

for k in kata:
    if len(k) < len(terpendek):
        terpendek = k
    if len(k) > len(terpanjang):
        terpanjang = k

print(f"terpendek: {terpendek}, terpanjang: {terpanjang}")
```

4.2 PENJELASAN

Pada soal diminta untuk menemukan kata terpendek dan terpanjang dalam sebuah kalimat yang dimasukkan oleh user.

```
kalimat = input("Kalimat: ")
```

Pertama, pada line pertama meminta user untuk memasukkan sebuah kalimat melalui fungsi `input()` dengan prompt "Kalimat: ". Kalimat yang dimasukkan oleh pengguna kemudian disimpan dalam variabel `kalimat`.

```
kata = kalimat.split()
```

Selanjutnya, ekspresi `kalimat.split()` digunakan untuk memisahkan kalimat menjadi sebuah daftar kata-kata. Fungsi `split()` secara default memisahkan string berdasarkan spasi, sehingga setiap kata dalam kalimat akan menjadi elemen-elemen dalam daftar `kata`.

```
terpendek = terpanjang = kata[0]
```

kemudian menginisialisasi variabel `terpendek` dan `terpanjang` dengan kata pertama dalam daftar `kata`, sehingga pada awalnya kata pertama akan dianggap sebagai kata terpendek dan terpanjang.

```
for k in kata:
    if len(k) < len(terpendek):
        terpendek = k
    if len(k) > len(terpanjang):
        terpanjang = k
```

Dalam loop `for` selanjutnya, akan memeriksa setiap kata dalam daftar `kata`. Untuk setiap kata, akan dibandingkan panjangnya (jumlah karakter) dengan panjang kata terpendek dan terpanjang yang telah ditemukan sebelumnya. Jika panjang kata tersebut lebih pendek dari `terpendek`, maka kata tersebut akan menjadi kata terpendek yang baru. Jika panjang kata tersebut lebih panjang dari `terpanjang`, maka kata tersebut akan menjadi kata terpanjang yang baru.

```
print(f"terpendek: {terpendek}, terpanjang: {terpanjang}")
```

Setelah loop selesai, kemudian akan di print hasilnya dalam format yang telah ditentukan, yaitu dengan menampilkan kata terpendek dan terpanjang yang ditemukan dalam kalimat. Format outputnya adalah "`terpendek: <kata_terpendek>, terpanjang: <kata_terpanjang>`".

4.3 HASIL OUTPUT

```
Kalimat: red snakes and a black frog in the pool
terpendek: a, terpanjang: snakes
```

```
Kalimat: saya makan maka saya kenyang
terpendek: saya, terpanjang: kenyang
```