



Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	71220876
Nama Lengkap	DEVINA ELISSE PUTRI
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.

MATERI 1


String adalah kumpulan karakter dalam program komputer yang disimpan sebagai satu kesatuan, mampu menyimpan huruf dalam kode ASCII. Meskipun tidak semua bahasa pemrograman memiliki tipe data String, ini adalah jenis data yang tidak dasar dan menyimpan lebih dari satu nilai tunggal sebagai satu kesatuan. Beberapa bahasa pemrograman menyebut String sebagai kumpulan karakter atau array/list of character.

MATERI 2

```
▶ 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'
```



String dapat diakses sebagai satu kesatuan dengan menyebut nama variabelnya atau per huruf dengan menyebutkan indeksinya. Indeks dimulai dari 0 dan harus berupa bilangan bulat. Pada memory komputer, string disimpan secara urut menggunakan list yang berisi huruf-huruf dengan indeks dimulai dari 0.

MATERI 3

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

  True
  True
  False

[4] if "saya" > "dia":
      print("Ya") #Ya
    else:
      print("Tidak")

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

  Ya
  Sama
```

FUNGSI LEN

Untuk mengetahui panjang sebuah string, gunakan operator `len(<string>)`. Untuk menampilkan huruf terakhir dari string, gunakan indeks string yang ke-`(len(<string>) - 1)`, karena indeks dimulai dari 0.

```
▶ 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'
```

```
↳ 42
a
a
t
```

TRANSFERSING STRING

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

```
↳ indonesia jaya
```

```
▶ kalimat = "indonesia jaya"
for kal in kalimat:
    print(kal,end='')

indonesia jaya
```

STRING SLICE

```
▶ 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
```

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

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()

OPERATOR * DAN + PADA STRING

Pada Python, operator + bisa menggabungkan dua string, sementara operator * dapat menampilkan string sejumlah perkaliannya.

```

▶ 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

```

```

↳ saya makan
   ulangulangulang
   ulang ulang

```

MATERI 4

Parsing string adalah metode menelusuri string untuk mendapatkan atau mengubah bagian yang diinginkan. Contoh, dari kalimat "Saudara-saudara, pada tanggal 17-08-1945 Indonesia merdeka", kita ingin mendapatkan tanggal bulan tahun dan mengubahnya menjadi format 08/17/1945. Langkah-langkah parsing string dapat dilakukan sebagai berikut:

1. Memisahkan string menjadi token berdasarkan spasi: "Saudara-saudara", "pada", "tanggal", "17-08-1945", "Indonesia", dan "merdeka".
2. Mencari token yang diawali dengan angka, kemudian memisahkan angka tersebut menggunakan pemisah '-'.
 17-08-1945 → 17-08-1945
3. Mengatur ulang token-token dari langkah sebelumnya untuk mencapai format yang diinginkan.

```

▶ 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])

```

```

08/17/1945

```

KEGIATAN PRAKTIKUM

1. Buatlah program untuk menghitung berapa huruf hidup pada sebuah kalimat per kata

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()

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

```
1
2
3
3
4
```

2. Buatlah program untuk membuat mengubah susunan format tanggal dari YYYY-MMDD menjadi DD-MM-YYYY

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

```
01-12-2020
```

3. Buatlah program untuk mengetahui apakah suatu kalimat adalah palindrom atau bukan! Palindrom adalah kalimat yang jika dibalik sama saja. Misalnya: Step on no pets, Pull up If I pull up, Some men interpret nine memos, dan Madam, In Eden I'm Adam.

```
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")
```

```
palindrom
```

4. Buatlah program untuk mengambil beberapa kata dari suatu kalimat yang diinputkan. Misal kalimat: "saya akan pergi sekolah". Akan diambil 1 kata, maka hasilnya: "saya", "akan", "pergi", "sekolah". Sedangkan jika 2 kata maka hasilnya: "saya akan", "akan pergi", "pergi sekolah".

Untuk mengambil beberapa kata dari kalimat, digunakan logika sebagai berikut:

- Ubah kalimat menjadi huruf kecil semua (lowercase)
- Pecah (split), kalimat dipisah berdasarkan spasi
- Untuk setiap kata yang sudah dipisah tersebut, maka ambil indeks kata per n buah, misal 1 buah jika n=1, 2 jika n=2. Indeks yang diambil adalah dimulai dari 0, jika n=2 maka ambil 0 dan 1, lalu ambil 1 dan 2, 2 dan 3, 3 dan 4, dst...dilakukan dalam perulangan.

```
#Kalimat
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))
```

```
['a quick', 'quick brown', 'brown fox', 'fox jumps', 'jumps over', 'over the', 'the lazy', 'lazy dog.', 'dog.']
```

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 1

1. `kata1 = kata1.lower().replace(" ", "")` dan `kata2 = kata2.lower().replace(" ", "")`: Baris ini bertujuan untuk mengubah kedua kata menjadi huruf kecil dan menghapus spasi. Hal ini dilakukan agar tidak ada perbedaan antara huruf besar dan huruf kecil dalam perbandingan.
2. `if len(kata1) != len(kata2): return False`: Fungsi memeriksa apakah panjang kedua kata sama. Jika panjangnya berbeda, berarti kata-kata tersebut tidak bisa menjadi anagram karena jumlah hurufnya tidak sama. Sehingga fungsi langsung mengembalikan `False`.
3. `sorted_kata1 = sorted(kata1)` dan `sorted_kata2 = sorted(kata2)`: Baris ini mengurutkan huruf-huruf dalam kedua kata menggunakan fungsi `sorted()`. Hal ini dilakukan agar kedua kata memiliki urutan huruf yang sama untuk membandingkan apakah keduanya adalah anagram.

4. `if sorted_kata1 == sorted_kata2: return True`: Fungsi memeriksa apakah kedua kata memiliki urutan huruf yang sama setelah diurutkan. Jika urutan hurufnya sama, maka kedua kata tersebut merupakan anagram, sehingga fungsi mengembalikan `True`.

5. `else: return False`: Jika urutan huruf kedua kata tidak sama setelah diurutkan, maka kedua kata tersebut bukan anagram, sehingga fungsi mengembalikan `False`.

```
def cek_anagram(kata1, kata2):
    # Mengubah kedua kata menjadi huruf kecil dan menghapus spasi
    kata1 = kata1.lower().replace(" ", "")
    kata2 = kata2.lower().replace(" ", "")

    # Memeriksa apakah panjang kedua kata sama
    if len(kata1) != len(kata2):
        return False

    # Mengurutkan huruf dalam kedua kata
    sorted_kata1 = sorted(kata1)
    sorted_kata2 = sorted(kata2)

    # Memeriksa apakah kedua kata memiliki urutan huruf yang sama
    if sorted_kata1 == sorted_kata2:
        return True
    else:
        return False

# Contoh penggunaan
kata1 = input("Masukkan kata pertama: ")
kata2 = input("Masukkan kata kedua: ")

if cek_anagram(kata1, kata2):
    print(f"{kata1} dan {kata2} adalah anagram.")
else:
    print(f"{kata1} dan {kata2} bukan anagram.")
```

```
Masukkan kata pertama: tama
Masukkan kata kedua: atma
tama dan atma adalah anagram.
```

SOAL 2

```
def hitung_frekuensi_kata(kalimat, kata):  
    # Menghitung jumlah kemunculan kata dalam kalimat  
    count = kalimat.count(kata)  
    return count  
  
# Meminta pengguna untuk memasukkan kalimat  
kalimat = input("Masukkan kalimat: ")  
  
# Meminta pengguna untuk memasukkan kata yang ingin dihitung kemunculannya  
kata_yang_dihitung = input("Masukkan kata yang ingin dihitung kemunculannya: ")  
  
# Menghitung frekuensi kemunculan kata  
frekuensi = hitung_frekuensi_kata(kalimat, kata_yang_dihitung)  
  
# Menampilkan output  
print(f"{kata_yang_dihitung} muncul sebanyak {frekuensi} kali dalam kalimat yang dimasukkan.")  
|  
} Masukkan kalimat: aku cinta kamu cinta dia juga cinta aku tidak cinta dia  
Masukkan kata yang ingin dihitung kemunculannya: cinta  
cinta muncul sebanyak 4 kali dalam kalimat yang dimasukkan.
```

1. `count = kalimat.count(kata)`: Baris ini menggunakan metode `count()` untuk menghitung berapa kali kata yang diberikan (`kata`) muncul dalam kalimat yang diberikan (`kalimat`). Metode `count()` mengembalikan jumlah kemunculan kata tersebut dalam kalimat.

2. `return count`: Fungsi mengembalikan nilai `count`, yaitu jumlah kemunculan kata yang dihitung.

Setelah mendefinisikan fungsi `hitung_frekuensi_kata`, program kemudian meminta pengguna untuk memasukkan sebuah kalimat dan kata yang ingin dihitung kemunculannya. Kemudian, program memanggil fungsi `hitung_frekuensi_kata` dengan kalimat dan kata yang dimasukkan oleh pengguna, dan menyimpan jumlah kemunculan kata tersebut dalam variabel `frekuensi`. Terakhir, program mencetak jumlah kemunculan kata tersebut sebagai output.

SOAL 3

```
def hapus_spasi_berlebih(kalimat):  
    # Memisahkan string menjadi kata-kata menggunakan spasi sebagai pemisah  
    kata_kalimat = kalimat.split()  
  
    # Menggabungkan kembali kata-kata tersebut dengan satu spasi sebagai pemisah  
    kalimat_baru = ' '.join(kata_kalimat)  
  
    return kalimat_baru  
  
# Meminta pengguna untuk memasukkan sebuah kalimat  
kalimat = input("Masukkan kalimat: ")  
  
# Menghapus spasi berlebih  
kalimat_tanpa_spasi_berlebih = hapus_spasi_berlebih(kalimat)  
  
# Menampilkan output  
print("Kalimat setelah menghapus spasi berlebih:")  
print(kalimat_tanpa_spasi_berlebih)  
|  
  
Masukkan kalimat: aku      cinta kamu  
Kalimat setelah menghapus spasi berlebih:  
aku cinta kamu
```

1. `kata_kalimat = kalimat.split()`: Baris ini menggunakan metode `split()` untuk memisahkan string kalimat menjadi kata-kata menggunakan spasi sebagai pemisah. Hasilnya adalah sebuah list yang berisi kata-kata.
2. `kalimat_baru = ' '.join(kata_kalimat)`: Baris ini menggunakan metode `join()` untuk menggabungkan kembali kata-kata tersebut menjadi sebuah string, dengan satu spasi sebagai pemisah antar kata.
3. `return kalimat_baru`: Fungsi mengembalikan string `kalimat_baru` yang sudah tidak memiliki spasi berlebih.

SOAL 4

```
def kata_terpendek_terpanjang(kalimat):
    # Memisahkan kalimat menjadi kata-kata menggunakan spasi sebagai pemisah
    kata_kalimat = kalimat.split()

    # Menginisialisasi kata terpendek dan terpanjang dengan kata pertama dalam kalimat
    kata_terpendek = kata_kalimat[0]
    kata_terpanjang = kata_kalimat[0]

    # Mencari kata terpendek dan terpanjang
    for kata in kata_kalimat:
        if len(kata) < len(kata_terpendek):
            kata_terpendek = kata
        if len(kata) > len(kata_terpanjang):
            kata_terpanjang = kata

    return kata_terpendek, kata_terpanjang

# Meminta pengguna untuk memasukkan sebuah kalimat
kalimat = input("Masukkan kalimat: ")

# Menggunakan fungsi untuk mendapatkan kata terpendek dan terpanjang
kata_terpendek, kata_terpanjang = kata_terpendek_terpanjang(kalimat)

# Menampilkan output
print(f"Kata terpendek: {kata_terpendek}")
print(f"Kata terpanjang: {kata_terpanjang}")
```

Masukkan kalimat: aku sangat mencintaimu
Kata terpendek: aku
Kata terpanjang: mencintaimu

1. `kata_kalimat = kalimat.split()`: Baris ini menggunakan metode `split()` untuk memisahkan string kalimat menjadi kata-kata menggunakan spasi sebagai pemisah. Hasilnya adalah daftar kata yang disimpan dalam variabel `kata_kalimat`.

2. `kata_terpendek = kata_kalimat[0]` dan `kata_terpanjang = kata_kalimat[0]`: Baris ini menginisialisasi kata terpendek dan kata terpanjang dengan kata pertama dalam kalimat.

3. Iterasi melalui semua kata dalam `kata_kalimat`:

- Jika panjang kata saat ini lebih pendek dari panjang `kata_terpendek`, maka kata tersebut akan disimpan sebagai `kata_terpendek`.
- Jika panjang kata saat ini lebih panjang dari panjang `kata_terpanjang`, maka kata tersebut akan disimpan sebagai `kata_terpanjang`.

4. Fungsi mengembalikan nilai `kata_terpendek` dan `kata_terpanjang`.