Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71230970**
Nama Lengkap	**Gregorius Daniel Jodan Perminas**
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.

Pengertian String di Python

Ada banyak tipe data yang tersedia di Python dan dapat digunakan. Salah satu yang cukup populer untuk digunakan adalah tipe data String. String adalah sekumpulan karakter atau huruf yang menjadi satu kesatuan dan digunakan oleh program komputer untuk menyimpan kalimat. Data string di Python biasanya disimpan dalam bentuk kode ASCII.

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	`
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[END OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	|
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(	88	58	1011000	130	X					
41	29	101001	51	)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[					
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	-	93	5D	1011101	135	]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	_					

Table 1 - ASCII

Pengaksesan dan Manipulasi String

String dapat dibuat dan disimpan dalam sebuah variabel. Objek string dalam variabel tersebut biasanya dibatasi oleh tanda kutip satu atau kutip dua, jadi semua karakter di dalam tanda kutip, baik itu berupa huruf, angka, atau karakter unik, merupakan bagian dari string tersebut.

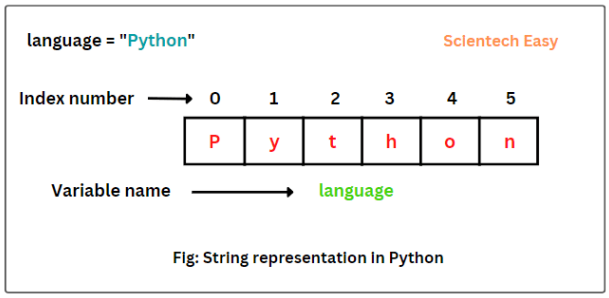
```python
namasaya = "Antonius Rachmat C"
temansaya1 = "Yuan Lukito"
temansaya2 = 'Laurentius Kuncoro'
temansaya3 = "Matahari" + 'Bakti'

print(temansaya3)
print(namasaya[9])
huruf = temansaya2[0]
print(huruf)
```

```
MatahariBakti
R
L
```

Gambar 1 - Pengaksesan String dalam Python

String dapat diakses seperti data lainnya dengan menyebut nama variabelnya, atau per huruf dengan menyebutkan indeksnya. Indeks string dimulai dari [0] seperti pada list. Namun ada catatan yang perlu diingat bahwa semua karakter yang berada di dalam list baik itu berupa huruf, angka, spasi, karakter unik atau apapun akan dimasukkan ke dalam indeks. Maka pada gambar di atas semisal saat dilakukan print(namasaya[9]), huruf ke 9 harusnya adalah ”a” namun yang dicetak justru ”R”, hal ini dikarenakan spasi dihitung juga dalam indeks.

Pada memory komputer, string disimpan secara urut menggunakan list yang berisi huruf/karakter/angka dengan indeks yang dimulai dari nol.



Gambar 2 - Penyimpanan String dalam Memory (Scientech Easy)

Operator dan Metode String

OPERATOR IN

Operator ”in” digunakan untuk mengecek apakah suatu kata/kalimat merupakan substring dari kata/kalimat lainnnya.

```python
kalimat = "saya mau makan"
print("mau" in kalimat)
print("dia" in kalimat)
```

```
True
False
```

*Gambar 3 - Penggunaan Operator **in** pada String*

OPERATOR COMPARISON

String juga dapat dibandingkan menggunakan operator comparison yang akan menghasilkan *output* True atau False.

```
1   if "abb" > "baa":
2       print("Yes!")
3   else:
4       print("No")
5
6   if "dua" == "dua":
7       print("Sama")
```

```
No
Sama
```

*Gambar 4 - Penggunaan **Operator Comparison** pada String*

FUNGSI len

Len digunakan untuk mengetahui berapa panjang (jumlah karakter) sebuah String. Bila kita ingin menampilkan karakter terakhir dari sebuah string maka kita harus menggunakan indeks len(<string>-1), karena indeks string dimulai dari [0].

```
1   kalimat = "Siapa yang cita-citanya jadi anime adik-adik?"
2   print(len(kalimat))
3   terakhir = kalimat[len(kalimat)-1]
4   print(terakhir)
```

```
45
?
```

*Gambar 5 - Penggunaan **len** pada String*

TRAVERSING STRING

Ini adalah metode untuk menampilkan semua karakter yang berada di dalam string satu-persatu. Hal ini dilakukan menggunakan *loop* pada Python dengan dua cara:

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

Gambar 6 - Dengan Akses Terhadap Indeks

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

Gambar 7 - Tanpa Akses Terhadap Indeks Secara Otomatis

String Slice

String slice adalah proses menampilkan substring (sebagian dari string) pada sebuah string dengan menggunakan indeks dari suatu awal yang ditentukan sampai akhir-1 yang ditentukan. Syntax yang

digunakan biasanya <string>[start:stop:step]. Jika start tidak diisi maka Python akan mengganggap slicing dimulai dari indeks pertama yaitu [0], demikian juga berlaku ketika stop tidak diisi maka Python akan menganggap slicing berakhir ketika indeks terakhir dari suatu string. Step di sini opsional.

```
kalimat = "cerita rakyat"
print(kalimat[7:len(kalimat)])
print(kalimat[:5])
print(kalimat[5:])
```

```
rakyat
cerit
a rakyat
```

Gambar 8 - Slicing String pada Python

CATATAN: String bersifat *immutable* yang artinya bahwa data tersebut tidak bisa diubah saat program berjalan, hanya bisa diinisialisasi saja. Contohnya:

```
kalimat = "satu"
kalimat[0] = "batu"
```

```
Traceback (most recent call last):
  File "d:\UKDW\Semester 2 - 2024\PraktikumAlpro_A_Dida\ProgramHere\coba2.py", line 2, in <module>
    kalimat[0] = "batu"
    ~~~~~~~^^^
TypeError: 'str' object does not support item assignment
```

Gambar 9 - Terjadi Error

Untuk mengubah string ketika program sudah berjalan, maka harus disimpan dalam variabel yang berbeda

```
kalimat = "satu"
kalimat_baru = kalimat[0] + "alah"
print(kalimat_baru)
```

```
salah
```

Gambar 10 - Tidak Terjadi Error

Berikut ini beberapa Method String yang dapat digunakan di Python:

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()

Table 2 - String Method dari Modul

Operator * dan + pada String

Pada Python operator + dan * bisa digunakan pada String. Operator + yang biasanya digunakan untuk menjumlahkan bilangan bisa digunakan untuk menggabungkan dua buah string. Operator * yang bisa digunakan untuk mengkalikan bilangan bisa digunakan untuk menampilkan string sejumlah perkaliannya

```
kata1 = "saya"
kata2 = "makan"
kata3 = kata1 + " " + kata2
print(kata3)
kata4 = "ulang"
print(kata4 * 4)
```

```
saya makan
ulangulangulangulang
```

*Gambar 11 – Penggunaan Operator * dan + Pada String*

Parsing String

Parsing string adalah cara menelusuri string bagian demi bagian untuk mendapatkan dan mengubah bagian string yang diinginkan. Misal ada kalimat seperti ini:

"Saudara-saudara, pada tanggal 17-08-1945 Indonesia merdeka"

Dari string di atas, kita ingin untuk mengambil tanggal bulan dan tahun lalu formatnya disusun ulang menjadi 08/17/1945. Maka program Pythonnya akan menjadi seperti 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])
```

```
08/17/1945
```

Gambar 12 - Program Untuk Parsing String

Cara kerja dari program di atas adalah:

- Melakukan split string berdasarkan spasi sehingga menjadi token: "Saudara-saudara,", "pada", "tanggal", "17-08-1945", "Indonesia", "merdeka".
- Lakukan perulangan untuk mencari token yang diawali dengan angka, kemudian lanjutkan dengan split angka tersebut dengan pemisah '-'.
- Kemudian susun ulang token-token dari langkah sebelumnya menjadi format yang diharapkan.

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.

LINK GITHUB: https://github.com/danieljodan/PrakAlPro7_A_71230970.git

SOAL 1

```
1   # Program Anagram
    Comment Code
2   def cekAnagram (masukan1, masukan2):
3       kata1 = masukan1.lower()
4       kata1Urut = sorted (kata1)
5       kata2 = masukan2.lower()
6       kata2Urut = sorted (kata2)
7       if kata1Urut == kata2Urut:
8           print(f"{kata1} dan {kata2} adalah anagram.")
9       else:
10          print("Bukan Anagram")
11
12  kata1 = input("Masukkan kata pertama: ")
13  kata2 = input("Masukkan kata kedua: ")
14  cekAnagram (kata1,kata2)
```

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

Gambar 13 - Program Untuk Latihan Mandiri 7.1

Program di atas digunakan untuk mengecek apakah kata pertama dan kedua yang di *input* oleh *user* merupakan anagram atau bukan. Penjelasan per baris:

Baris 1 = Comment

Baris 2 = Pendefinisian fungsi cekAnagram()

Baris 3,5 = Mengubah semua huruf ke lowercase

Baris 4,6 = Mengurutkan huruf supaya dapat di *compare*

Baris 7-8 = Jika kedua kata yang sudah diurutkan ternyata sama maka print bahwa kata1 dan kata2 adalah anagram

Baris 9-10 = Jika kedua kata tidak sama maka print bukan anagram

Baris 12-13 = Menerima *input* kata dari *user*

Baris 14 = Memanggil fungsi dengan *input user* sebagai argumen

SOAL 2

```python
# Program Menghitung Frekuensi Kemunculan Kata pada Kalimat
def hitungKata(kalimat, kata):
    kalimatLower = kalimat.lower()
    kataLower = kata.lower()
    kalimatLowerBersih = "".join([i for i in kalimatLower if i.isalpha()])
    frekuensi = kalimatLowerBersih.count(kataLower)
    return frekuensi

kalimat = input("Masukkan kalimat Anda: ")
kata = input("Masukkan kata yang jumlahnya akan dicari: ")
print (f"Kata {kata} ada {hitungKata(kalimat,kata)} buah")
```

```
Masukkan kalimat Anda: Saya mau makan. Makan itu wajib. Mau siang atau malam saya wajib makan
Masukkan kata yang jumlahnya akan dicari: makan
Kata makan ada 3 buah
```

Gambar 14 - Program Untuk Latihan Mandiri 7.2

Program ini digunakan untuk menghitung frekuensi kemunculan sebuah kata dari sebuah kalimat.
Penjelasan kode:

Baris 1 = Comment

Baris 2 = Pendefinisian fungsi hitungKata()

Baris 3 = Mengubah kalimat yang *diinput user* menjadi kalimat dalam *lowercase*

Baris 4 = Mengubah kata yang *diinput user* menjadi kata dalam *lowercase*

Baris 5 = Membuang semua karakter yang bukan alphabet dan menyimpan hasilnya dalam variabel

Baris 6 = Menghitung frekuensi kemunculan kata dalam *string*

Baris 7 = Melakukan *return* ke fungsi

Baris 9-10 = Menerima *input* kalimat dan kata yang ingin dicari oleh *user*

Baris 11 = Melakukan print dan pemanggilan fungsi dengan *input user* sebagai argumen

SOAL 3

```python
# Program Menghapus Spasi Berlebih
def spasiBerlebih(kalimat):
    kalimatBersih1 = " ".join(kalimat.split())
    return kalimatBersih1

masukan1 = input("Masukkan kalimat:\n")
print("Kalimat dengan spasi normal:\n"+spasiBerlebih(masukan1))
```

```
Masukkan kalimat:
   saya tidak suka   memancing ikan
Kalimat dengan spasi normal:
saya tidak suka memancing ikan
```

Gambar 15 - Program Untuk Latihan Mandiri 7.3

Program ini digunakan untuk menghapus semua spasi berlebih pada sebuah kalimat dan menjadikannya sebagai kalimat dengan spasi normal. Penjelasan kode:

Baris 1 = Comment

Baris 2 = Pendefinisian fungsi spasiBerlebih()

Baris 3 = Memisahkan kalimat menjadi beberapa token kemudian menggabungnya dengan fungsi join ke dalam sebuah variabel

Baris 4 = Melakukan return nilai pada fungsi

Baris 6 = Menerima *input* dari *user*

Baris 7 = Melakukan print dan pemanggilan fungsi dengan *input user* sebagai argumen

SOAL 4

```python
# Program Untuk Mengecek Kata Terpendek dan Terpanjang
def kataPendekPanjang(kalimat):
    kata = kalimat.split()
    terpendek = min(kata, key=len)
    terpanjang = max(kata, key=len)
    print(f"Kata terpendek adalah {terpendek}")
    print(f"kata terpanjang adalah {terpanjang}")

masukan1 = input("Masukkan kalimat: ")
kataPendekPanjang(masukan1)
```

```
Masukkan kalimat: red snakes and a black frog in the pool
terpendek: a
terpanjang: snakes
```

Gambar 16 - Program Untuk Latihan Mandiri 7.4

Program ini dapat menampilkan kata terpendek dan terpanjang dari suatu kalimat. Penjelasan kode:

Baris 1 = Comment

Baris 2 = Pendefinisian fungsi kataPendekPanjang()

Baris 3 = Memisah kalimat menjadi beberapa token

Baris 4 = Digunakan untuk mencari kata pendek dalam list kata. Fungsi min() mengembalikan item dengan nilai minimum, dan argumen key=len berarti bahwa pemilihan dilakukan berdasarkan panjang kata.

Baris 5 = Digunakan untuk mengembalikan item dengan nilai maksimum, dan argumen key=len berarti bahwa pemilihan dilakukan berdasarkan panjang kata.

Baris 6 = Mencetak kata terpendek

Baris 7 = Mencetak kata terpanjang

Baris 9 = Menerima *input* dari *user*

Baris 10 = Memanggil fungsi dengan *input user* sebagai argumen