



Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	<ISI DENGAN NIM ANDA>
Nama Lengkap	<ISI DENGAN NAMA LENGKAP ANDA>
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

Penjelasan materi 1

➤ 7.3.1 Pengantar String

String adalah untaian karakter-karakter yang menjadi satu kesatuan dan digunakan dalam program komputer untuk menyimpan kalimat, baik panjang ataupun pendek. String adalah suatu jenis data yang mampu menyimpan huruf / karakter dan disimpan dalam kode ASCII. Tidak semua bahasa pemrograman memiliki tipe data String, seperti misalnya bahasa C. Tipe data ini merupakan jenis tipe data yang bukan tipe data dasar, karena tipe data ini pada dasarnya menyimpan lebih dari satu nilai tunggal sebagai satu kesatuan.

➤ 7.3.2 Pengaksesan String dan Manipulasi String

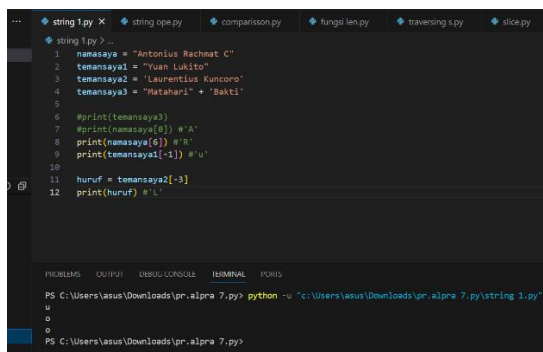
Deklarasi Variabel String: Variabel string dapat dibuat dengan mendeklarasikan variabel dan langsung mengisinya dengan data menggunakan tanda kutip tunggal atau ganda .

Akses String: Anda dapat mengakses string sebagai satu kesatuan dengan menyebutkan nama variabelnya atau per huruf dengan menyebutkan indeksinya. Indeks pada string dimulai dari 0, mirip dengan list.

Indeks String: Indeks string harus berupa bilangan bulat dan tidak boleh pecahan.

Penyimpanan di Memory: String disimpan secara urut dalam memori komputer menggunakan list yang berisi huruf-huruf dengan indeks dimulai dari nol.

Dengan demikian, Anda dapat membuat, mengakses, dan memanipulasi string dengan menggunakan variabel dan indeksinya sesuai kebutuhan dalam Python.



```
1. namanaya = "Antonius Rachmat C"
2. temansaya1 = "Yun Lukito"
3. temansaya2 = "Laurentius Kuncoro"
4. temansaya3 = "Matahari" + "Bakti"
5.
6. #print(temansaya3)
7. #print(temansaya[0]) # 'A'
8. print(temansaya[5]) # 'n'
9. print(temansaya[-1]) # 'u'
10.
11. huruf = temansaya2[-3]
12. print(huruf) # 'o'
```

Output:

```
PS C:\Users\asus\Downloads\pr_alpra 7.py> python -u "c:\Users\asus\Downloads\pr_alpra 7.py\string 1.py"
o
o
o
PS C:\Users\asus\Downloads\pr_alpra 7.py>
```

1. Membuat String: String dibuat dengan menetapkan kumpulan karakter ke dalam variabel menggunakan tanda kutip tunggal (') atau ganda ("). Penggabungan string dapat dilakukan dengan menggunakan operator `+`.

2. Akses String:

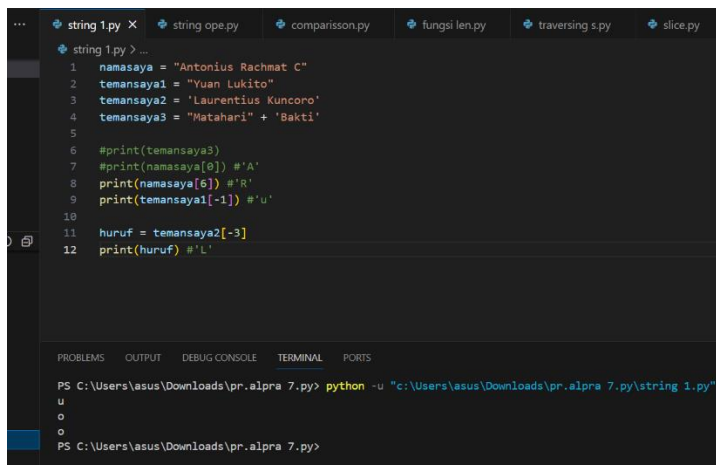
- String dapat diakses sebagai satu kesatuan dengan menyebutkan nama variabelnya.
- Untuk mengakses karakter individual, gunakan indeks yang dimulai dari 0.

3. Indeks dalam String:

- Indeks string harus berupa bilangan bulat, bukan pecahan.

➤ 7.3.3 Operator dan Metode String OPERATOR in

kita menggunakan operator in untuk memeriksa apakah suatu string merupakan substring dari string lain. Operator ini menghasilkan nilai True jika substring ditemukan, dan False jika tidak. Contohnya:



```
string 1.py X string ope.py comparison.py fungsi len.py traversing s.py slice.py
string 1.py > ...
1  namesaya = "Antonius Rachmat C"
2  temansaya1 = "Yuan Lukito"
3  temansaya2 = 'Laurentius Kuncoro'
4  temansaya3 = "Matahari" + 'Bakti'
5
6  #print(temansaya3)
7  #print(namesaya[0]) #'A'
8  print(namesaya[6]) #'R'
9  print(temansaya1[-1]) #'u'
10
11 huruf = temansaya2[-3]
12 print(huruf) #'L'
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\asus\Downloads\pr.alpra 7.py> python -u "c:\Users\asus\Downloads\pr.alpra 7.py\string 1.py"
u
o
o
PS C:\Users\asus\Downloads\pr.alpra 7.py>
```

Selain operator in, kita juga bisa melakukan perbandingan langsung antara dua string menggunakan operator perbandingan (>, <, ==, dll.), yang akan menghasilkan nilai True atau False berdasarkan urutan leksikal dari string tersebut:

Dengan demikian, Anda dapat membuat, mengakses, dan memanipulasi string dengan mudah menggunakan variabel dan indeksinya dalam Python.

➤ FUNGSI len

Program Python tersebut menggunakan fungsi `len()` untuk mengetahui panjang sebuah string dan indeks untuk mengakses karakter tertentu dalam string. Penggunaan indeks positif dimulai dari 0, sedangkan penggunaan indeks negatif dimulai dari belakang. Ini memungkinkan kita untuk dengan mudah mengakses karakter terakhir dari sebuah string menggunakan `kalimat[-1]`.

```
(variable) kalimat: Literal['universitas kristen duta wacana yogyakarta']
fungsi
1  kalimat = "universitas kristen duta wacana yogyakarta"
2  print(len(kalimat))
3
4  terakhir = kalimat[len(kalimat)-1]
5  print(terakhir)
6
7  #bisa juga menggunakan indeks -1
8  terakhir_versi2 = kalimat[-1]
9  print(terakhir_versi2)
10 #atau menggunakan indeks -2 untuk huruf terakhir kedua
11 terakhir2 = kalimat[-2]
12 print(terakhir2)
13
14

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\asus\Downloads\pr.alpra 7.py> python -u "c:\Users\asus\Downloads\pr.alpra 7.py"
42
a
a
t
PS C:\Users\asus\Downloads\pr.alpra 7.py>
```

➤ TRAVERSING STRING

Program-program Python di atas menampilkan string "indonesia jaya" huruf demi huruf. Program pertama menggunakan loop while dengan akses terhadap indeks, sedangkan program kedua menggunakan loop for tanpa akses terhadap indeks.

```
(variable) kalimat: Literal['universitas kristen duta wacana yogyakarta']
fungsi
1  kalimat = "universitas kristen duta wacana yogyakarta"
2  print(len(kalimat))
3
4  terakhir = kalimat[len(kalimat)-1]
5  print(terakhir)
6
7  #bisa juga menggunakan indeks -1
8  terakhir_versi2 = kalimat[-1]
9  print(terakhir_versi2)
10 #atau menggunakan indeks -2 untuk huruf terakhir kedua
11 terakhir2 = kalimat[-2]
12 print(terakhir2)
13
14

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\asus\Downloads\pr.alpra 7.py> python -u "c:\Users\asus\Downloads\pr.alpra 7.py"
42
a
a
t
PS C:\Users\asus\Downloads\pr.alpra 7.py>
```

Program-program Python di atas bertujuan untuk menampilkan string "indonesia jaya" huruf demi huruf. Program pertama menggunakan loop `while` dengan akses terhadap indeks, sedangkan program kedua menggunakan loop `for` tanpa akses terhadap indeks. Pada program pertama, variabel `i` digunakan sebagai indeks untuk mengakses karakter dalam string `kalimat` dengan menggunakan loop `while` hingga `i` mencapai panjang string.

Pada program kedua, loop `for` langsung mengiterasi melalui setiap karakter dalam string `kalimat`, tanpa perlu menggunakan indeks tambahan. Kedua program mencapai tujuan yang sama, yaitu menampilkan string "indonesia jaya" huruf demi huruf. Namun, program kedua lebih ringkas karena menggunakan loop `for` tanpa memerlukan penggunaan variabel indeks tambahan.

➤ STRING SLICE

String slicing di Python memungkinkan pemotongan substring menggunakan sintaks `string[awal:akhir]`, dengan opsi untuk mengosongkan bagian awal atau akhir. String bersifat immutable, artinya tidak dapat diubah setelah dibuat, tetapi bisa dibuat variabel baru untuk menyimpan hasil modifikasi.

```
traversing s.py > ...
1  kalimat = "indonesia jaya"
2  i = 0
3  while i < len(kalimat):
4      print(kalimat[i])
5      i += 1
6
7
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
n
e
s
i
a

j
a
y
a
```

PS C:\Users\asus\Downloads> python3 traversing s.py

➤ 7.4 Kegiatan Praktikum

```
mat.py string 1.py string ope.py comparisson.py
tugas tes.py > hitung_huruf_hidup
1  def hitung_huruf_hidup(kalimat):
2      huruf_hidup = 'aeiouAEIOU'
3      jumlah_huruf_hidup = 0
4      for kata in kalimat.split():
5          for huruf in kata:
6              if huruf in huruf_hidup:
7                  jumlah_huruf_hidup += 1
8      return jumlah_huruf_hidup
```

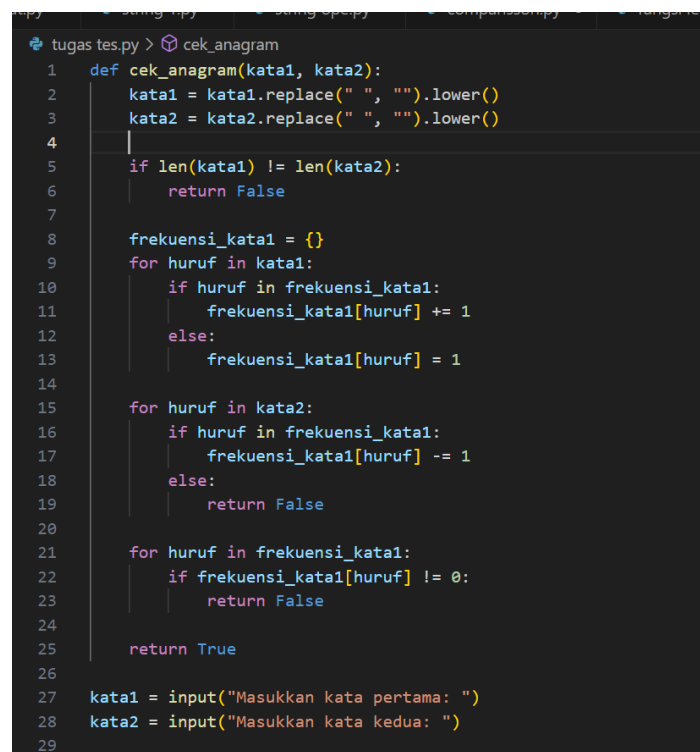

Program ini meminta pengguna untuk memasukkan sebuah kalimat dan memberikan keluaran apakah kalimat tersebut adalah palindrom atau tidak. Dengan catatan ini, diharapkan pemahaman tentang setiap kasus menjadi lebih jelas.

MATERI 2

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

The image shows a screenshot of a code editor with a dark background. At the top, there are tabs for 'tugas tes.py', 'cek_anagram', and 'komparasi.py'. The main window displays a Python script for checking anagrams. The script defines a function 'cek_anagram(kata1, kata2)' that takes two strings as input. It first replaces spaces with empty strings and converts both to lowercase. Then, it checks if the lengths are equal. If not, it returns False. If yes, it creates a frequency dictionary for the first string and iterates through the second string, updating the frequency counts. If any character in the second string is not in the first or has a higher frequency, it returns False. Finally, it checks if all frequency counts in the dictionary are greater than zero. If not, it returns False. Otherwise, it returns True. Below the function, there are two input prompts: 'Masukkan kata pertama: ' and 'Masukkan kata kedua: '.

```
1 def cek_anagram(kata1, kata2):
2     # Menghapus spasi dan mengubah semua huruf menjadi huruf kecil
3     kata1 = kata1.replace(" ", "").lower()
4     kata2 = kata2.replace(" ", "").lower()
5
6     # Cek apakah panjang kedua kata sama
7     if len(kata1) != len(kata2):
8         return False
9
10    # Menghitung frekuensi setiap huruf dalam kata1
11    frekuensi_kata1 = {}
12    for huruf in kata1:
13        if huruf in frekuensi_kata1:
14            frekuensi_kata1[huruf] += 1
15        else:
16            frekuensi_kata1[huruf] = 1
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
PS C:\Users\asus\Downloads\pr.alpra 7.py> python -u "c:\Users\asus\Downloads\pr.alpra 7.py\tugas tes.py"
Masukkan kata pertama: 5
Masukkan kata kedua: 10
'5' dan '10' bukan anagram.
PS C:\Users\asus\Downloads\pr.alpra 7.py> python -u "c:\Users\asus\Downloads\pr.alpra 7.py\tugas tes.py"
Masukkan kata pertama: 2
Masukkan kata kedua: 2
'2' dan '2' adalah anagram.
PS C:\Users\asus\Downloads\pr.alpra 7.py> python -u "c:\Users\asus\Downloads\pr.alpra 7.py\tugas tes.py"
Masukkan kata pertama: 10
```

Fungsi `cek_anagram` menerima dua string (`kata1` dan `kata2`) sebagai input.

Pertama, program menghapus semua spasi dari kedua kata dan mengubahnya menjadi huruf kecil untuk memastikan perbandingan yang case-insensitive dan tidak terpengaruh oleh spasi.

Program kemudian memeriksa apakah kedua kata memiliki panjang yang sama; jika tidak, mereka tidak bisa menjadi anagram.

Selanjutnya, program menghitung frekuensi masing-masing huruf di kata pertama dan mengurangi frekuensi berdasarkan huruf-huruf di kata kedua.

Jika kedua kata adalah anagram, frekuensi setiap huruf di akhir perbandingan akan menjadi nol.

Terakhir, program akan memberikan output bahwa kedua kata tersebut adalah anagram atau bukan berdasarkan hasil perbandingan frekuensi hurufnya.

SOAL 2



```
traversing s.py  tugas tes.py  slice.py  s3.py  opr.py  parsing string.py
soal 4.py > ...
1  text = '"Saya mau makan malam.'"
2
3  def ambil_kata_kalimat(kalimat, n):
4      kalimat = kalimat.lower()
5      print(kalimat)
6      hasil_akhir = []
7      hasil = kalimat.split()
8      print(hasil)
9      for i in range (0,len(hasil)):
10         tmp = ' '.join(hasil[i:i + n])
11         hasil_akhir.append(tmp)
12
13     return hasil_akhir
14
15 print(ambil_kata_kalimat(text, 3))
16
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
PS C:\Users\asus\Downloads\pr.alpra 7.py> python -u "c:\Users\asus\Downloads\pr.alpra 7.py\soal 4.py"
"saya mau makan malam."
["saya", "mau", "makan", "malam."]
["saya mau makan", "mau makan malam.", "makan malam.", "malam."]
PS C:\Users\asus\Downloads\pr.alpra 7.py>
```

Fungsi `hitung_kemunculan_kata` menerima dua parameter: `kalimat` (string yang akan dianalisis) dan `kata` (kata yang akan dicari frekuensinya).

Pada fungsi tersebut, kita pertama-tama mengubah kedua input menjadi huruf kecil menggunakan metode `lower()`.

Selanjutnya, kita menggunakan metode `count()` untuk menghitung berapa kali kata muncul dalam kalimat.

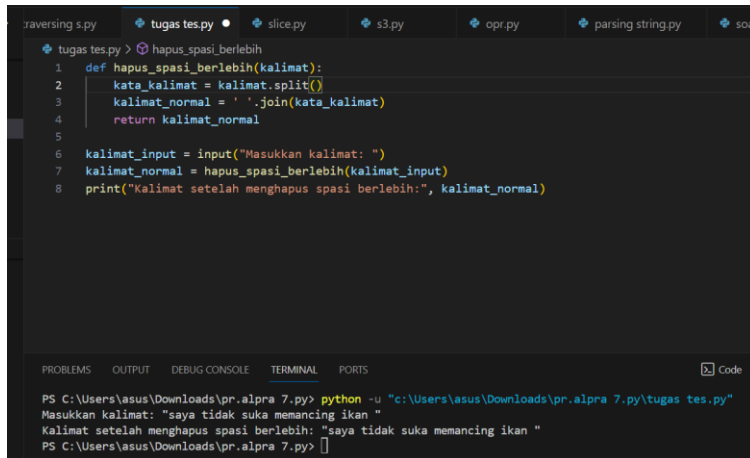
Fungsi mengembalikan jumlah kemunculan kata tersebut dalam kalimat.

Kemudian, program meminta pengguna untuk memasukkan kalimat dan kata yang ingin dihitung frekuensinya.

Hasilnya dicetak sebagai output.

Dengan ini, program akan menghitung dan memberikan jumlah kemunculan kata yang diminta dalam string yang diberikan.

SOAL 3



```
traversing s.py  tugas tes.py  slice.py  s3.py  opr.py  parsing string.py  soal

tugas tes.py > hapus_spasi_berlebih
1 def hapus_spasi_berlebih(kalimat):
2     kata_kalimat = kalimat.split()
3     kalimat_normal = ' '.join(kata_kalimat)
4     return kalimat_normal
5
6 kalimat_input = input("Masukkan kalimat: ")
7 kalimat_normal = hapus_spasi_berlebih(kalimat_input)
8 print("Kalimat setelah menghapus spasi berlebih:", kalimat_normal)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\asus\Downloads\pr.alpra 7.py> python -u "c:\Users\asus\Downloads\pr.alpra 7.py\tugas tes.py"
Masukkan kalimat: "saya tidak suka memancing ikan "
Kalimat setelah menghapus spasi berlebih: "saya tidak suka memancing ikan "
PS C:\Users\asus\Downloads\pr.alpra 7.py> []
```

Fungsi `hapus_spasi_berlebih` menerima string kalimat sebagai input.

Pertama, program membagi string tersebut menjadi kata-kata menggunakan method `split()`, yang secara default memisahkan string berdasarkan spasi.

Setelah itu, program menggabungkan kembali kata-kata tersebut dengan satu spasi sebagai pemisah menggunakan method `join()`.

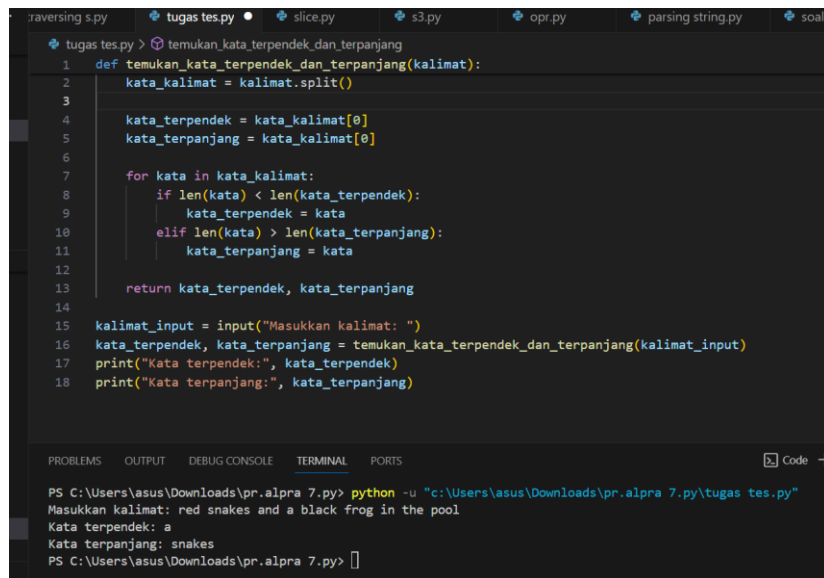
Kemudian, fungsi mengembalikan string yang telah diubah menjadi satu spasi normal.

Program kemudian meminta pengguna untuk memasukkan kalimat.

Hasilnya dicetak sebagai output.

Dengan ini, program akan menghapus semua spasi berlebih dalam string yang diberikan dan mengembalikan string tersebut dengan satu spasi normal.

SOAL 4



```
traversing s.py  tugas tes.py  slice.py  s3.py  opr.py  parsing string.py  soal 4
tugas tes.py > temukan_kata_terpendek_dan_terpanjang
1 def temukan_kata_terpendek_dan_terpanjang(kalimat):
2     kata_kalimat = kalimat.split()
3
4     kata_terpendek = kata_kalimat[0]
5     kata_terpanjang = kata_kalimat[0]
6
7     for kata in kata_kalimat:
8         if len(kata) < len(kata_terpendek):
9             kata_terpendek = kata
10        elif len(kata) > len(kata_terpanjang):
11            kata_terpanjang = kata
12
13    return kata_terpendek, kata_terpanjang
14
15 kalimat_input = input("Masukkan kalimat: ")
16 kata_terpendek, kata_terpanjang = temukan_kata_terpendek_dan_terpanjang(kalimat_input)
17 print("Kata terpendek:", kata_terpendek)
18 print("Kata terpanjang:", kata_terpanjang)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\asus\Downloads\pr.alpra 7.py> python -u "c:\Users\asus\Downloads\pr.alpra 7.py\tugas tes.py"
Masukkan kalimat: red snakes and a black frog in the pool
Kata terpendek: a
Kata terpanjang: snakes
PS C:\Users\asus\Downloads\pr.alpra 7.py>
```

Fungsi `temukan_kata_terpendek_dan_terpanjang` menerima string kalimat sebagai input.

Pertama, program membagi kalimat menjadi kata-kata menggunakan `method split()`.

Kemudian, program menginisialisasi kata terpendek dan terpanjang dengan kata pertama dari kalimat.

Program kemudian melakukan iterasi melalui setiap kata dalam kalimat.

Selama iterasi, jika panjang kata lebih pendek dari kata terpendek yang saat ini, kata terpendek akan diperbarui.

Jika panjang kata lebih panjang dari kata terpanjang yang saat ini, kata terpanjang akan diperbarui.

Setelah selesai iterasi, program mengembalikan kata terpendek dan terpanjang.

Kemudian, program meminta pengguna untuk memasukkan kalimat.

Kata terpendek dan terpanjang dicetak sebagai output.