Laporan Praktikum Algoritma dan Pemrograman

Semester Genap 2023/2024

NIM	**71230982**
Nama Lengkap	**Rainie Fanita Chrisabel Hadisantoso**
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

MATERI : Pengolahan String

Materi dari pertemuan ke 7 ini adalah tentang string, Tujuan dri pembelajaran pertemuan ke 7 ini adalah menjelaskan string, mengakses dan memanipulasi string, menggunakan operator string, dan melakukan parsing string sederhana.

Pengantar String

String merupakan gabungan karakter-karakter menjadi satu kesatuan data. Biasanya string diguakan utnuk menyimpan data berupa huruf, simbol, dan kalimat. Karakter-karakter yang dipakai dalam string merupakan karakter dengan standar ASCII. Secara sederhana string sama saja dengan kumpulan tipe data seperti array atau list, karena itu string bukan salah satu tipe data dasar. String bukan merupakan tipe data dasar karena tipe data dasar menampung nilai-nilaidata sebagai satu kesatuan, sedangkan string bertindak sebagai array. Oleh sebab itu, tidak semua bahasa pemrograman memiliki tipe data string, contohnya bahasa pemrograman C yang tidak memiliki string tapi memiliki array dan list.

Pengaksesan String dan Manipulasi String

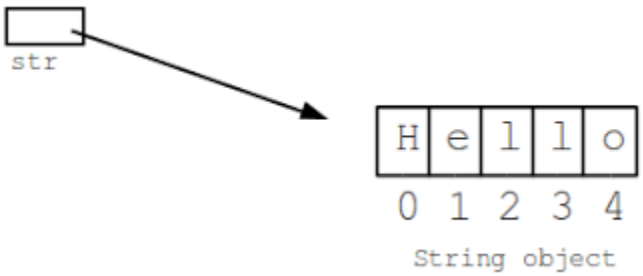
String dapat dibuat dengan cara menggunakan variabel seperti biasa: mendeklarasikan nama variabel dan langsung diisi dengan data di antara 2 tanda petik (“”).

Untuk mengakses list, ada dua metode yang dapat dilakukan:

1. Diakses sebagai satu kesatuan dengan cara memanggil nama variabelnya. Cukup menggunakan perintah print() dan menyebutkan nama stringnya. Hasil source code dibawah ini adalah: Hello, World!.

```
quote = "Hello, World!"
print(quote)
```

2. Diakses secara per karakter dengan cara memanggil nama dengan indeksnya. Cara penulisannya adalah: `string[i]`. Dimana nilai i adalah indeks karakter yang hendak kita panggil. Konsep indeks dalam string sama dengan indeks list, indeksi harus berupa bilangan bulat dan indeks dimulai dari 0 hingga jumlah karakter-karakter dalam string – 1. Indeks dimulai dari 0 karena data dalam string disimpan dalam memory dengan cara seperti itu juga. Berikut penggambaran peletakan indeks dalam string.



Gambar 1. Indeks String

Selain urut dari 0 hingga panjang string, indeks dalam string juga bisa menggunakan bilangan negatif. Indeks negatif, disebut juga dengan reverse indeks, digunakan ketika kita ingin mengakses string

dengan urutan dari akhir string. Karakter terakhir string memiliki indeks -1, kedua terakhir -2 dan seterusnya hingga mencapai nilai jumlah karakter secara negatif.

Index		0	1	2	3	4	5	6	7	8	9	10	11
Character		H	e	l	l	o		W	o	r	l	d	!
Reverse Index		−12	−11	−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

Gambar 2. Indeks dan Reverse Indeks

Contoh penulisan:

```
quote = "Hello, World!"

print(quote[0])      #H
print(quote[1])      #e
print(quote[-1])     #!
print(quote[-13])    #H
```

String “Hello, World!” memiliki panjang 13 karakter, karena itu indeks ke -13 adalah “H”, sama dengan indeks ke 0.

Operator dan Metode String

Ada banyak operator untuk string dalam Python. Operator-operator tersebut biasanya digunakan untuk mengakses dan memanipulasi data dalam sebuah string. Berikut tabel beberapa operator yang dapat digunakan dalam Python.

Operator	Description	Example
+	Concatenation - Adds values on either side of the operator	strA + strB will give Hello World!
*	Repetition - Creates new strings, concatenating multiple copies of the same string	strA * 2 will give -Hello Hello
[]	Slice - Gives the character from the given index	strA[1] will give e
[:]	Range Slice - Gives the characters from the given range	strA[1:4] will give ell
in	Membership - Returns true if a character exists in the given string	H in strA will give 1
not in	Membership - Returns true if a character does not exist in the given string	M not in strA will give 1
r/R	Raw String - Suppresses actual meaning of Escape characters.	print r'\n' prints \n and print R'\n' prints \n
%	Format - Performs String formatting	It's the same like we used in C.

Gambar 3. Tabel Operator String

Ada banyak operator untuk mengakses dan memanipulasi string dala Pyhon.

1. Operator in:

   Operator in berfungsi mencari bagian dalam sebuah string. Dengan operator string kita dapat mencari huruf, angka, pokoknya karakter dalam string tersebut. Apabila karakter yang dicari ada dalam string, maka operator akan mengembalikan nilai True. Apabila karakter tidak ada dalam string, maka operator akan mengembalikan False. Karakter atau bagian dalam sebuah string biasa disebut juga dengan substring.

```
quote = "Hello, World!"
word = "World"

print(word in quote)    #True
print("l" in quote)     #True
print("L" in quote)     #False
```

   Dari contoh source code diatas, dapat dilihat bahwa variabel word ada dalam variabel quote. Word berisi kata “World” sehingga menghasilkan True. Kemudian dalam perintah print “l” in quote bernilai True, karena terdapat 3 huruf l dalam kalimat “Hello, World!”. Namun dalam print selanjutnya, “L” in quote menghasilkan nilai False, padahal ada huruf l dalam kalimat tersebut. Hal tersebut terjadi karena string dibaca secara case sensitif, sehingga karakter “l” dan karakter “L” dibedakan.

2. Operator comparison:

   Kita juga dapat menggunakan operator comparisor (> ,< ,==,!=) untuk string. Operator comparison dalam string berdasarkan urutan dan nilai karakter dari kode ASCII atau Unicode. Karena itu case sensitif juga berlaku dalam operator comparison. Karakter huruf besar memiliki nilai kode yang lebih kecil dari karakter huruf kecil. Penulisan operator comparison untuk string sama dengan penulisan untuk nilai angka.

```
quote = "Hello, World!"

print(quote == "Hello, World!")     #True
print(quote == "hello, world!")     #False
print("A" < "a")                    #True
print("R" > "A")                    #True
```

   Sama seperti hasil comparison bilangan, comparison string juga akan menampilkan nilai True dan Fale ketika kondisi comparison tidak dipenuhi oleh string.

3. Operator * dan + pada string:

   Dalam Python operator pertambahan (+) dan operator perkalian (*) dapat digunakan utnuk mengolah string.

   - Operator pertambahan (+): Dapat menambahkan dua buah string menjadi satu. Namun tidak dapat menambahkan dua tipe data yang berbeda menjadi satu.

```
quote = "Hello, World!"
word = "World"
```

```
print(word + quote)      #World Hello, World!
print(word + 4)          #ERROR
```

Jika string ditambahkan dengan tipe data lain seperti int, maka program akan menampilkan ERROR.

- Operator perkalian (*): Dapat menampilkan string berkali-kali sesuai dengan bilangan bulat pengali. Namun tidak dapat mengalikan string dengan string lainnya atau bilangan float.

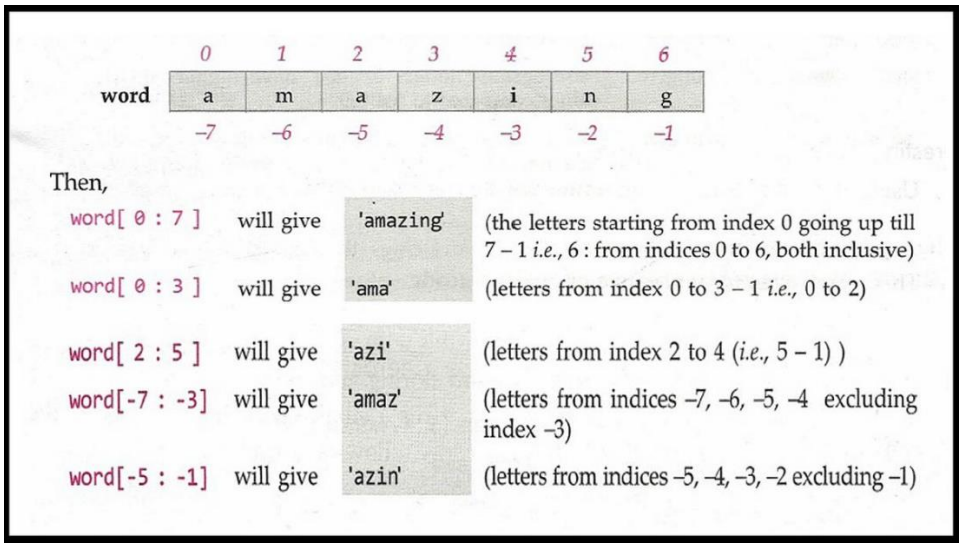
```
word = "World"

print(word * 4)          #WorldWorldWorldWorld
print(word * quote)      #ERROR
```

Word dikalikan dengan 4 akan menghasilkan output World sebanyak 4 kali. Namun jika string dikalikan dengan string, maka program akan menampilkan ERROR.

4. String slice:

String slice (:) berfungsi untuk menampilkan substring atau bagian dalam string berdasarkan indeks awal tertentu sampai indeks akhir tertentu. Penulisan string slice adalah: `string[i start:i stop]`. Berikut beberapa contoh string slice:



Gambar 4. Contoh String Slice

Dapat dilihat akan mengambil substring dari 0 sampai 7, sehingga hasilnya “amazing”. String slice akan mengambil substring dari indeks awal hingga indeks akhir. Berlaku juga dengan reverse indeks, akan diambil substring berdasarkan reverse indeks. Jika kita hanya memasukkan indeks awal atau indeks akhir, maka otomatis substring yang diambil adalah sisa indeks dari indeks awal hingga terakhir. Namun jika kita memasukkan indeks awal lebih besar dari indeks akhir, maka string slice tidak akan menampilkan apa-apa.

Metode String

Ada banyak metode yang dapat digunakan untuk string dalam Python. Metode-metode tersebut biasanya digunakan untuk mengakses dan memanipulasi data-data dalam string. Salah satu contohnya adalah fungsi metode len.

1. Fungsi len:

   Fungsi len digunakan untuk menemukan panjang data dalam string dengan cara menghitung jumlah total karakter-karakter di dalam string. Penulisan fungsi len adalah: `len(string)` untuk menghitung jumlah seluruh karakter string. Fungsi len jug dapat digunakan untuk menampilkan substring dengan indeks reverse, dengan penulisan: `len(string - i).`

   ```
   quote = "Hello, World!"

   print(len(quote))       #13
   print(len(quote)-1)     #!
   ```

   Fungsi len banyak digunakan untuk perulangan berbasis jumlah string, yaitu traversing string.

2. Traversing string:

   Traversing string adalah cara untuk menampilkan substring demi substring dalam sebuah string dengan menggunakan loop. Untuk menampilkan menggunakan loop ada dua cara yang menghasilkan hasil yang sama:

   - Menggunakan indeks menggunakan fungsi len untuk mencari panjang string sebagai end perulangan.

     ```
     quote = "Hello, World!"
     i = 0
     while i < len(quote):
         print(quote[i],end='')
         i += 1
     ```

   - Menggunakan indeks secara otomatis.

     ```
     quote = "Hello, World!"

     for word in quote:
         print(word, end='')
     ```

Selain fungsi len, ada berbagai macam metode lainnya, berikut tabel beberapa perintah metode yang sering digunakan dalam praktikum:

Method	Description
strip()	removes any whitespace from the beginning or the end
lower()	Returns a string in lower case characters
upper()	Returns a string in uppercase characters
replace()	Replaces a string with another string
split()	Splits the string into sub strings
capitalize()	Capitalizes the first character in the string
count()	Returns no. of occurrences in the string
index()	Returns the index of the character
find()	Gives the index value of the string specified
isalpha()	Returns true if the string has only alphabets
isalnum()	Returns true if the string has both alphabets and numbers
isdigit()	Returns true if the string has only numbers
islower()	Returns true if the string has only lower case characters
isupper()	Returns true if the string has only uppercase characters

Gambar 5. Tabel Metode String

Dan masih ada banyak lagi metode string dalam Python, yang dapat dilihat dalam library Python berikut: https://docs.python.org/library/stdtypes.html#string-methods

Parsing String

Parsing string adalah metode untuk menemukan atau mendapatkan substring yang diinginkan sehingga dapat diubah dan dimanipulasi sesuai keinginan. Parsing string dilakukan dengan menggunakan berbagai metode yang ada di dalam list diatas. Berikut contoh parsing string dengan string tanggal = "hari ini tanggal 22-2-2022" dan ingin mengganti "-" dengan "/" maka:

```
tanggal = "hari ini tanggal 22-2-2022"

tanggal = "/".join(tanggal.split("-"))
print(tanggal)      #hari ini tanggal 22/2/2022
```

Digunakan 2 metode bertingkat yaitu join() dan split(). Fungsi split() digunakan untuk memisahkan substring dalam tanggal berdasarkan tanda "-". Kemudian substring yang dipisahkan digabungkan kembali oleh fungsi join() dengan tanda "/". Sehingga hasil menjadi "hari ini tanggal 22/2/2022".

LATIHAN MANDIRI 7

SOAL 7.1

Membuat program untuk mendeteksi apakah 2 kata yang dimasukkan adalah anagram.

a. Source code

```
[ ]  def anagram(kata1, kata2):
       if sorted(kata1.lower()) and sorted(kata2.lower()):
         print(f"{kata1} dan {kata2} adalah anagram")
       else:
         print(f"{kata1} dan {kata2} bukan anagram")

     kata1 = input("Masukkan kata 1 = ")
     kata2 = input("Masukkan kata 2 = ")

     anagram(kata1, kata2)
```

b. Output

```
Masukkan kata 1 = mata
Masukkan kata 2 = atma
mata dan atma adalah anagram
```

c. Penjelasan

Untuk menghitung apakah kedua kata merupakan anagram, digunakan:

- Lower(): membuat semua kata dalam bentuk huruf kecil. Komputer membaca string secara case sensitive sehingga kata yang sama dengan penggunaan case yang berbeda dibedakan. Dengan lower, semua kata otomatis huruf keci, sehingga komputer membaca huruf dalam kata sama.
- Sorted(): mengurutkan huruf dan simbol secara urutan ASCII. Huruf-huruf dalam kedua kata diurutkan untuk memudahkan identifikasi anagram.

Ketika kedua kata dalam huruf kecil dan sudah diurutkan, jika urutan huruf kata pertama dan kedua sama maka anagram. Jika tidak maka bukan anagram.

SOAL 7.2

Membuat program untuk mencari jumlah kata dalam sebuah kalimat.

a. Source code

```
def hitung(kalimat, kata) :
    kalimat = kalimat.lower()
    kata = kata.lower()
    hasil = kalimat.count(kata)
    return hasil

kalimat = input("Masukkan kalimat : ")
kata = input("Masukkan kata : ")

print("Jumlah kata", kata, "adalah", hitung(kalimat,kata))
```

b. Output

```
Masukkan kalimat : Saya mau makan. Makan itu wajib. Mau siang atau malam saya wajib makan
Masukkan kata : makan
Jumlah kata makan adalah 3
```

c. Penjelasan

- Digunakan lower() untuk string kalimat dan string kata agar semuanya berupa huruf kecil. Komputer membaca string secara case sensitive sehingga kapitalisasi string perlu diubah secara uppercase atau lowercase agar terbaca sama.
- Kemudian ditambah variabel baru untuk menghitung jumlah kata dengan perintah count(). Count berfungsi untuk mendeteksi string dalam kalimat dengan susuanan kata yang sama dengan string kata dan menghitung jumlahnya.

SOAL 7.3

Program untuk membetulkan spasi sebuah kalimat.

a. Source code

```
[4]  def spasi(kalimat):
       bener = " ".join(kalimat.split())
       bener = bener.strip()
       print(bener)

     kalimat= input("Masukkan kalimat = ")
     print("Hasil pembetulan spasi = ", end= "")
     spasi(kalimat)
```

b. Output

```
Masukkan kalimat = saya tidak suka memancing ikan
Hasil pembetulan spasi = saya tidak suka memancing ikan
```

c. Penjelasan

- Pertama string kalimat di split() agar dibagi menjadi tiap kata tanpa spasi. Kemudian digunakan join() agar string-string yang dipisah kembali disatukan dengan “ “ atau spasi.
- Lalu digunakan strip() untuk menghilangkan spasi diawal maupu dibelakang kalimat.

SOAL 7.4

Program untuk mencari kata dengan jumlah huruf terpanjang dan terpendek.

a. Source code

```
[5]  def hitung(kalimat):
       pisah = kalimat.split()

       terpendek = min(pisah, key=len)
       terpanjang = max(pisah, key=len)

       print("Kata terpendek = ", terpendek)
       print("Kata terpanjang = ", terpanjang)

     kalimat = input("Masukkan kalimat = ")

     hitung(kalimat)
```

b. Output

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

c. Penjelasan

- Variabel pisah untuk menampung string yang dipisah-pisah berdasarkan spasi dengan menggunakan split(). Dilakukan agar tiap kata dalam kalimat terpisah menjadi string sendiri-snediri.
- Variabel terpendek untuk mencari kata terpendek menggunakan perintah min().Perintah min(),digunakan untuk menentukan panjang string terkecil. String-string dalam pisah dihitung panjangnya dengan perintah key = len atau bentuk lainnya len().
- Variabel terpanjang untuk mencari kata terpanjang menggunakan max(). String dalam pisah dihitung panjangnya menggunakan key = len, kemudian string dengan nilai terbesar dipanggil oleh max().

Github Repository:

https://github.com/rainiefch/Praktikum-Algoritma-Pemrograman_71230982.git