MODUL PRAKTIKUM PEMROSESAN BAHASA ALAMI (OPERASI STRING PADA PEMROSESAN BAHASA ALAMI)

Bambang Pilu Hartato, S.Kom., M.Eng.

FAKULTAS ILMU KOMPUTER | INFORMATIKA

A. Tujuan Praktikum

Praktikum kali ini membahas beberapa operasi yang cukup penting dalam Pemrosesan Bahasa Alami, seperti Operasi String, Tokenisasi, Lemmatisasi, dan Pengukuran Jarak Levenshtein. Setelah mengikuti praktikum kali ini diharapkan kita dapat melakukan operasi-operasi tertentu yang dapat digunakan untuk melakukan pemrosesan bahasa alami.

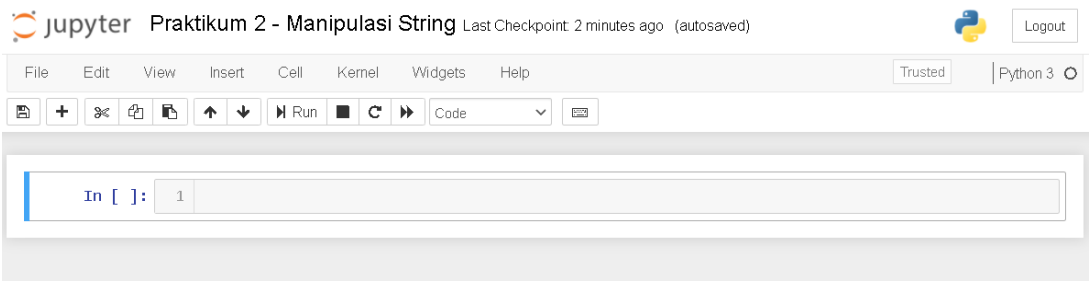
B. Pembahasan

Seperti yang telah dijelaskan sebelumnya, bahwa salah satu tantangan dalam pemrosesan bahasa alami adalah tidak terstrukturnya bahasa yang digunakan oleh manusia. Agar dapat diolah atau dipahami oleh mesin maka bahasa tersebut perlu dimanipulasi sedemikian rupa. Pada praktikum kali ini, kita akan belajar untuk melakukan beberapa operasi tertentu yang cukup sering digunakan dalam pemrosesan bahasa alami. Diasumsikan bahwa bahasa yang digunakan oleh manusia dituangkan ke dalam bentuk teks. Sehingga di praktikum kali ini, seluruh operasi yang dilakukan pada praktikum kali ini berkaitan dengan teks/string.

Untuk memulai praktikum kali ini, mari kita lakukan instruksi-instruksi berikut.

1. Buat File Notebook Baru pada Jupyter Notebook

Mari kita buka Jupyter Notebook kita pada active directory kita masing masing, misalnya saja pada "D:\Belajar PBA". Buat file Python 3 dan beri nama "Praktikum 2 – Manipulasi String". Gambar 1 menunjukkan file baru yang telah kita buat.



Gambar 1 File Baru

2. Membuat String Sederhana

Dalam Python, tipe data string ditandai dengan tanda petik tunggal atau petik ganda yang mengapit teks, misalnya "ini adalah contoh string". Umumnya tipe data string disimpan pada sebuah variabel. Lakukan perintah seperti yang ditunjukkan oleh Gambar 2.

```
In [1]:
1 string_pba = "Ini adalah string sederhana."
2 print(string_pba)

Ini adalah string sederhana.
```

Gambar 2 String Sederhana

Gambar 2 menunjukkan contoh sebuah string yang disimpan pada variabel bernama `string_pba`. Anda dapat mengganti petik ganda tersebut dengan petik tunggal.

3. Penggunaan Quotation Mark

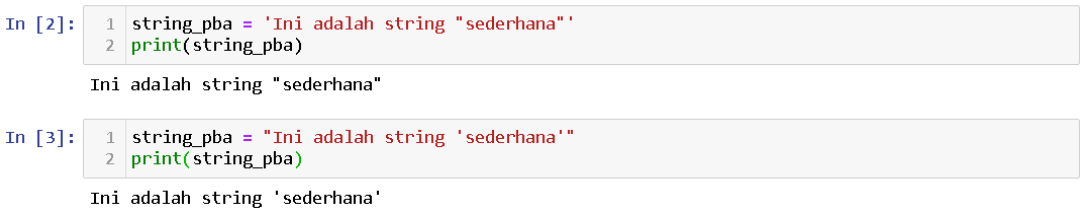
Pada kasus-kasus tertentu, ada kalanya kita menggunakan single quote (‘ ’) untuk membuat string, misalnya:

```
string_pba = ‘Ini adalah string sederhana’
```

Jika kita memutuskan akan melakukan quote pada kata ‘sederhana’ dengan cara berikut:

```
string_pba = ‘Ini adalah string ‘sederhana’’
```

maka program akan mengalami kesalahan. Untuk mengatasi masalah tersebut, kita dapat menggunakan dua opsi berikut:



Gambar 3 Contoh Penggunaan Quotation Mark

Gambar 3 menunjukkan penggunaan quotation mark untuk string pada Python. Terdapat du acara, kita dapat menggunakan salah satunya.

4. Mengambil Informasi dari String

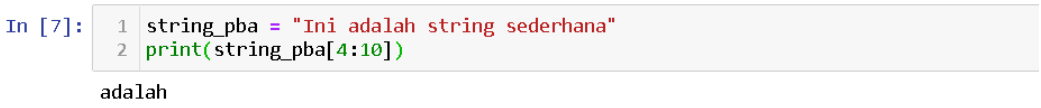
Pada kasus-kasus tertentu, terkadang kita ingin mengetahui beberapa informasi penting pada string yang kita gunakan. Berikut beberapa contoh operasi yang dapat dilakukan.

a. Mengakses karakter pada suatu indeks tertentu

Gambar 4 Mengakses Karakter pada Suatu Indeks

Gambar 4 menunjukkan contoh proses pengaksesan karakter pada suatu indeks. Pada contoh tersebut yang diakses adalah indeks ke-5 sehingga yang ditampilkan adalah karakter “d”. Perlu diingat bahwa indeks dimulai dari 0.

b. Mengakses substring



Gambar 5 Mengakses Substring

Gambar 5 menunjukkan contoh proses pengaksesan suatu substring pada string yang kita miliki. Pada contoh tersebut substring yang diakses ada pada indeks ke-4 sampai dengan indeks ke-9. Sehingga yang ditampilkan kata “adalah”. Teknik

tersebut dinamakan slicing. Adapun sintaks untuk melakukan slicing adalah sebagai berikut:

```
string[indeks_awal:indeks_akhir+1]
```

c. Menghitung panjang string

```
In [8]:
1 string_pba = "Ini adalah string sederhana"
2 pjg_string = len(string_pba)
3 print(pjg_string)

27
```

Gambar 6 Menghitung Panjang String

Gambar 6 menunjukkan contoh proses penghitungan panjang string. Untuk menghitung panjang sebuah string dapat menggunakan perintah `len(string)`.

d. Menghitung jumlah huruf tertentu pada string

```
In [11]:
1 string_pba = "Ini adalah string sederhana"
2 print("jumlah huruf 'a': %d" % (string_pba.count('a')))

jumlah huruf 'a': 5
```

Gambar 7 Menghitung Jumlah Karakter Tertentu

Gambar 7 menunjukkan proses penghitungan jumlah kemunculan karakter ‘a’ pada `string_pba`. Seperti yang terlihat, jumlah huruf ‘a’ pada string tersebut adalah 5, maka output dari proses tersebut adalah 5. Sintaks yang dapat digunakan untuk menghitung kemunculan huruf tertentu adalah

```
string.count(char)
```

e. Mencari indeks suatu huruf atau suatu kata

```
In [1]:
1 string_pba = "Ini adalah string sederhana"
2 print(string_pba.find('s'))

11
```

Gambar 8 Mencari indeks huruf s

Gambar 8 menunjukkan fungsi pencarian indeks suatu huruf. Jika kita perhatikan, output dari perintah tersebut adalah 11. Hal ini karena perintah find() hanya menampilkan indeks dari huruf ‘s’ yang muncul pertama kali.

Perintah find() juga dapat digunakan untuk pencarian kata seperti yang ditunjukkan pada Gambar 9.

```
In [2]:
1 string_pba = "Ini adalah string sederhana"
2 print(string_pba.find('adalah'))

4
```

Gambar 9 Mencari indeks kata adalah

5. Manipulasi String

Pada beberapa kasus, tidak jarang pemrosesan bahasa alami melibatkan proses manipulasi string. Berikut beberapa contoh proses manipulasi string yang umum digunakan.

a. Uppercase

Uppercase akan membuat semua karakter dari string yang dimanipulasi menjadi huruf kapital (uppercase). Untuk melakukan uppercase, dapat digunakan perintah `string.upper()`.

```
In [6]:
1 string_alay = "InI aDalaH StrIng ALay"
2 string_upper = string_alay.upper()
3
4 print("String sebelum uppercase: " + string_alay)
5 print("String sesudah uppercase: " + string_upper)
```

```
String sebelum uppercase: InI aDalaH StrIng ALay
String sesudah uppercase: INI ADALAH STRING ALAY
```

Gambar 10 Fungsi Uppercase

Gambar 10 menunjukkan salah satu contoh proses uppercase pada suatu string.

b. Lowercase

Lowercase akan membuat semua karakter dari string yang dimanipulasi menjadi huruf kecil (lowercase). Untuk melakukan lowercase, dapat digunakan perintah `string.lower()`

```
In [8]:
1 string_alay = "InI aDalaH StrIng ALay"
2 string_lower = string_alay.lower()
3
4 print("String sebelum lowercase: " + string_alay)
5 print("String sesudah lowercase: " + string_lower)
```

```
String sebelum lowercase: InI aDalaH StrIng ALay
String sesudah lowercase: ini adalah string alay
```

Gambar 11 Fungsi Lowercase

Gambar 11 menunjukkan salah satu contoh proses lowercase pada suatu string. Sebenarnya, untuk melakukan lowercase kita juga dapat menggunakan perintah `string,casefold()` seperti yang ditunjukkan oleh Gambar 12.

```
In [9]:
1 string_alay = "InI aDalaH StrIng ALay"
2 string_casefold = string_alay.casefold()
3
4 print("String sebelum casefold: " + string_alay)
5 print("String sesudah casefold: " + string_casefold)
```

```
String sebelum casefold: InI aDalaH StrIng ALay
String sesudah casefold: ini adalah string alay
```

Gambar 12 Fungsi Casefold

c. Title Case

Perintah ini akan membuat setiap kata dari string yang kita manipulasi diawali dengan huruf kapital dan diikuti huruf kecil. Untuk melakukan perintah ini dapat digunakan fungsi `string.title()`.

```
In [10]:
1 string_alay = "InI aDalaH StrIng ALay"
2 string_title = string_alay.title()
3
4 print("String sebelum title case: " + string_alay)
5 print("String sesudah title case: " + string_title)
```

```
String sebelum title case: InI aDalaH StrIng ALay
String sesudah title case: Ini Adalah String Alay
```

Gambar 13 Fungsi Title

Gambar 13 menunjukkan contoh penggunaan fungsi `title()`. Output yang dihasilkan oleh fungsi tersebut adalah string yang setiap katanya diawali oleh huruf kapital.

d. Capitilize

Perintah tersebut digunakan untuk memanipulasi string agar huruf pertama di awal string menjadi kapital.

```
In [12]:
1 string_alay = "InI aDalaH StrIng ALay."
2 string_capitalize = string_alay.capitalize()
3
4 print("String sebelum capitalize: " + string_alay)
5 print("String sesudah capitalize: " + string_capitalize)
```

```
String sebelum capitalize: InI aDalaH StrIng ALay.
String sesudah capitalize: Ini adalah string alay.
```

Gambar 14 Fungsi Capitalize

Gambar 14 menunjukkan contoh penggunaan fungsi `capitalize()`. Output yang dihasilkan akan membuat huruf pertama dari string menjadi huruf kapital.

e. Swapcase

Perintah swapcase digunakan untuk menukar case huruf. Sederhananya perintah ini akan menukar lower case menjadi upper case dan upper case menjadi lower case.

```
In [13]:
1 string_alay = "INI HURUF BESAR. ini huruf kecil."
2 string_swapcase = string_alay.swapcase()
3
4 print("String sebelum swapcase: " + string_alay)
5 print("String sesudah swapcase: " + string_swapcase)
```

```
String sebelum swapcase: INI HURUF BESAR. ini huruf kecil.
String sesudah swapcase: ini huruf besar. INI HURUF KECIL.
```

Gambar 15 Fungsi Swapcase

Gambar 15 menunjukkan contoh penggunaan fungsi `swapcase()`.

f. Reversed

Pada kasus-kasus tertentu kita membutuhkan proses pembalikan string. Untuk melakukan pembalikasn string dapat menggunakan perintah `string[::-1]` seperti yang ditunjukkan oleh

```
In [17]:
1 string_asli = "Universitas Amikom Purwokerto"
2
3 print("String asli: " + string_asli)
4 print("String reversed: " + string_asli[::-1])

String asli: Universitas Amikom Purwokerto
String reversed: otrekowruP mokimA satisrevinU
```

Gambar 16 Reversed String

g. Join

Perintah ini digunakan untuk menyisipkan karakter atau string tertentu di antara setiap karakter string yang sedang kita manipulasi.

```
In [19]:
1 string_asli = "Universitas Amikom Purwokerto"
2
3 print('-'.join(string_asli))

U-n-i-v-e-r-s-i-t-a-s- -A-m-i-k-o-m- -P-u-r-w-o-k-e-r-t-o
```

Gambar 17 Fungsi Join

Gambar 17 menunjukkan fungsi join. Pada contoh tersebut, dilakukan penyisipan karakter dash (-) pada setiap karakter `string_asli`.

6. Operasi Aritmetika pada String

Pada tipe data string, hanya dua operasi aritmetika yang berlaku, yaitu + dan *. Berikut penjelasannya.

a. Konkatenasi

Konkatenasi adalah operasi menyambungkan dua buah atau lebih string menjadi satu buah string. Untuk melakukan konkatenasi dapat menggunakan operator plus (+).

```
In [1]:
1 string_1 = "Ini adalah teks 1"
2 string_2 = "Ini adalah teks 2"
3
4 print(string_1 + " dan " + string_2)

Ini adalah teks 1 dan Ini adalah teks 2
```

Gambar 18 Operasi Konkatenasi

Gambar 18 menunjukkan contoh operasi konkatenasi terhadap `string_1` dan `string_2`.

b. Multiplikasi

Operasi multiplikasi adalah operasi untuk mencetak string berkali-kali. Untuk melakukan operasi ini dapat menggunakan operator asteris (*).

```
In [4]:
1 string_1 = "gandakan "
2
3 print(string_1 * 4)

gandakan gandakan gandakan gandakan
```

Gambar 19 Operasi Multiplikasi

Gambar 19 menunjukkan operasi multiplikasi yang dilakukan terhadap `string_1`.

7. Memeriksa Properti String

Pada keadaan tertentu, terkadang kita diharuskan untuk memeriksa karakteristik suatu string. Berikut beberapa operasi yang dapat dilakukan untuk melakukan hal tersebut.

a. isalnum()

Fungsi ini digunakan untuk memeriksa apakah suatu string hanya terdiri dari karakter alfanumerik. Fungsi ini akan memberikan jawaban True jika string hanya berisi karakter alabet dan numerik. Jika sebaliknya maka fungsi ini akan memberikan nilai False.

```
In [9]:
1 string_pba = "Amikom"
2
3 print(string_pba.isalnum())

True
```

Gambar 20 Operasi isalnum()

Gambar 20 menunjukkan proses penggunaan fungsi `isalnum()`. Fungsi tersebut memberikan nilai True karena `string_pba` tidak mengandung selain karakter alfabet dan numeris.

b. isalpha()

Fungsi ini digunakan untuk memeriksa apakah suatu string hanya terdiri dari karakter alfabet. Fungsi ini akan memberikan nilai True jika string hanya terdiri dari karakter alfabet. Sebaliknya jika string mengandung karakter selain alfabet makan fungsi ini akan memberikan nilai False.

```
In [12]:
1 string_pba = "Amikom 45"
2
3 print(string_pba.isalpha())

False
```

Gambar 21 Operasi isalpha()

c. isdigit()

Fungsi ini digunakan untuk memeriksa apakah suatu string hanya mengandung angka. Fungsi ini akan memberikan nilai True jika string hanya mengandung angka. Sebaliknya, jika string mengandung karakter selain angka maka fungsi ini akan memberikan nilai False.

```
In [14]:
1 string_pba = "12345"
2
3 print(string_pba.isdigit())

True
```

Gambar 22 Operasi isdigit()

Gambar 22 menunjukkan operasi `isdigit()` yang dilakukan terhadap `string_pba`. Output yang dihasilkan oleh fungsi tersebut adalah Trur. Hal tersebut dikarenakan string hanya mengandung angka.

8. Mengganti String

Pada kasus-kasus tertentu, terkadang kita diharuskan untuk melakukan penggantian sebagian atau keseluruhan string. Fungsi yang dapat digunakan untuk melakukan hal tersebut adalah `replace()`. Sintaks penggunaannya adalah sebagai berikut

```
string.replace(string_awal,string_pengganti)
```

```
In [23]:
1 string_pba = "STMIK Amikom Purwokerto"
2 print("String sebelum direplace: " + string_pba)
3 print("String sesudah direplace: " + string_pba.replace("STMIK","Universitas"))

String sebelum direplace: STMIK Amikom Purwokerto
String sesudah direplace: Universitas Amikom Purwokerto
```

Gambar 23 Operasi Replace

Gambar 23 menunjukkan proses penggantian kata STMIK menjadi Universitas pada `string_pba`.

9. Tokenisasi Kalimat

Tokenisasi kalimat adalah memecah string yang terdiri dari beberapa kalimat menjadi suatu list kalimat. Cara paling mudah untuk melakukan hal tersebut adalah dengan menggunakan fungsi `sent_tokenize()` yang ada pada library `nltk`.

```
In [25]:
1 import nltk
2
3 paragraf = '''The First sentence is about Python. The Second: about Django.
4 You can learn Python,Django and Data Ananlysis here.'''
5 token_kalimat = nltk.sent_tokenize(paragraf)
6
7 print(token_kalimat)

['The First sentence is about Python.', 'The Second: about Django.', 'You can learn Python,Django
and Data Ananlysis here.']
```

Gambar 24 Tokenisasi Kalimat

Gambar 24 menunjukkan contoh proses tokenisasi kalimat yang dilakukan oleh fungsi `sent_tokenize()`.

10. Tokenisasi Kata

Tokenisasi kata adalah memecah string yang terdiri dari beberapa kata menjadi suatu list kata. Cara paling mudah untuk melakukan hal tersebut adalah dengan menggunakan fungsi `word_tokenize()` yang ada pada library `nltk`.

```
In [26]:
1 import nltk
2
3 kalimat = 'It originated from the idea that there are readers who prefer learning new skills'
4 token_kata = nltk.word_tokenize(kalimat)
5
6 print(token_kata)

['It', 'originated', 'from', 'the', 'idea', 'that', 'there', 'are', 'readers', 'who', 'prefer',
'learning', 'new', 'skills']
```

Gambar 25 Tokenisasi Kata

Gambar 25 menunjukkan contoh proses tokenisasi kata yang dilakukan oelh fungsi `word_tokenize()`.

11. Stemming

Stemming adalah proses menghilangkan kata imbuhan dari suatu string dengan cara memotongnya.

```
In [37]:
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

ps = PorterStemmer()

kalimat = "Programmers program with programming languages"
token_kata = word_tokenize(kalimat)

for token in token_kata:
    print(token + " : " + ps.stem(token))
```

```
Programmers : programm
program : program
with : with
programming : program
languages : languag
```

Gambar 26 Proses Stemming

Gambar 26 menunjukkan proses stemming. Pada contoh tersebut stemming dilakukan untuk setiap kata yang ada di dalam string `kalimat`. Teknik stemming yang digunakan adalah teknik Porter's Stemmer.

12. Lemmatisasi

Lemmatisasi adalah proses untuk menemukan kata dasar dari suatu string dengan cara mencari akar katanya. Berbeda dengan stemming, lemmatisasi memperhatikan part of speech.

```
In [38]:
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print("rocks : " + lemmatizer.lemmatize('rocks'))
print("corpora : " + lemmatizer.lemmatize('corpora'))
print("better : " + lemmatizer.lemmatize('better', pos='a'))
```

```
rocks : rock
corpora : corpus
better : good
```

Gambar 27 Proses Lemmatisasi

Gambar 27 menunjukkan contoh proses lemmatisasi terhadap beberapa string. `WordNetLemmatizer()` secara default menggunakan parameter Part of Speech bernilai "noun".

13. Levenshtein Distance

Levenstein Distance atau sering disebut Minimum Edit Distance adalah suatu teknik untuk melihat kemiripan 2 buah string. Semakin kecil nilai distance nya maka semakin mirip 2 buah string tersebut. Untuk lebih jelas tentang teorinya, kita dapat mengakses video yang ada pada link berikut https://www.youtube.com/watch?v=JEQeF5gYeLc.

In [49]:

```
import nltk

string_1 = "kaktus"
string_2 = "takus"

print("Levenshtein Distance : %d" % (nltk.edit_distance(string_1,string_2)))
```

```
Levenshtein Distance : 2
```

Gambar 28 Levenshtein Distance

Gambar 28 menunjukkan contoh proses perhitungan Jarak Levenshtein antara kata “katus” dan “takus”. Dari hasil perhitungan yang dilakukan terlihat bahwa jarak yang terbentuk antara “katus” dan “takus” adalah 2.