Bab 9. Manipulasi String

Di bab-bab sebelumnya, sudah pernah disinggung tentang tipe data string. Khusus pada bab ini akan dibahas mengenai bagaimana cara mengolah atau memanipulasi data string.

Sebuah string di dalam Python dapat ditulis dengan diapit *single quotes* (tanda petik tunggal) maupun *double quote* (tanda petik ganda).

Kelebihan dari penggunaan tanda petik ganda adalah kita bisa menyisipkan tanda petik tunggal di dalam string tersebut. Contohnya

```
nama = "Siti 'Aisah"
print(nama)
```

```
Siti 'Aisah
```

Apabila kita menggunakan tanda petik tunggal untuk mengapit string dan kita ingin menyisipkan tanda petik tunggal juga di dalam stringnya, maka caranya dengan menambahkan karakter *backslash* di depan tanda petiknya. Contohnya

```
nama = 'Siti \'Aisah'
print(nama)
```

```
Siti 'Aisah
```

Escape Characters

Escape characters adalah karakter-karakter yang memiliki makna khusus. Tabel 9.1 menunjukkan beberapa di antaranya

Tabel 9.1 Daftar escape characters dalam Python

Karakter	Makna
\n	Baris baru (new line)
\t	Tab
\'	Single quote
\”	Double quote

Adapun berikut ini adalah contoh penggunaannya:

```
myString = "Hello\nWorld"
print(myString)
```

```
Hello
World
```

Apabila contoh tersebut diperhatikan, maka penambahan karakter \n di depan ‘World’akan berefek pada ganti barisnya string tersebut.

Perhatikan pula contoh berikut ini untuk penggunakan karakter \t yang memiliki makna sama dengan memberikan tab.

```
myString = "Hello\tWorld"
print(myString)
```

```
Hello	World
```

Operator `in` dan `not in` dalam String

Seperti halnya list, tuple, dan *dictionary*, untuk string juga bisa digunakan operator `in` dan `not in`. Operator ini digunakan untuk mengecek apakah suatu substring terdapat dalam string tersebut.

Contoh:

```
myString = 'Hello World, I\'m Fine'
print('World' in myString)
print('world' not in myString)
print('hello' in myString)
```

```
True
True
False
```

Dari contoh di atas, perintah `'World' in myString` untuk mengecek apakah string 'World' ada di dalam string `myString` atau tidak. Dalam hal ini ya, sehingga dihasilkan `True`. Sedangkan untuk perintah `'hello' in myString` akan dihasilkan nilai `False`, karena string 'hello' berbeda dengan 'Hello' yang ada dalam `myString`.

Perintah `upper()` dan `lower()`

Perintah `upper()` digunakan untuk mengubah string menjadi huruf kapital semuanya. Sedangkan `lower()` untuk mengubah string ke huruf kecil. Contohnya:

```
myString = 'Hello World, I\'m Fine'
print(myString.upper())
print(myString.lower())
```

```
HELLO WORLD, I'M FINE
hello world, i'm fine
```

Perintah `join()`

Selain menggunakan operator string *concatenation* (+), kita juga bisa melakukan penggabungan string dengan perintah `join()`. Kelebihan dari `join()` adalah kita bisa menentukan karakter yang menjadi penghubung antar string yang digabung. Contohnya adalah:

```
gabung = ', '.join(['mangga', 'apel', 'rambutan'])
print(gabung)

gabung = ' dan '.join(['mangga', 'apel', 'rambutan'])
print(gabung)

gabung = ' '.join(['mangga', 'apel', 'rambutan'])
print(gabung)
```

```
mangga, apel, rambutan
mangga dan apel dan rambutan
mangga apel rambutan
```

Perintah `gabung = ', '.join(['mangga', 'apel', 'rambutan'])` akan menggabungkan semua string `'mangga'`, `'apel'`, dan `'rambutan'`. Kemudian di antara stringnya terdapat string `', '` sehingga akan dihasilkan string `'mangga, apel, rambutan'`.

Hal yang sama juga berlaku untuk perintah `gabung = ' dan '.join(['mangga', 'apel', 'rambutan'])` dan `gabung = ' '.join(['mangga', 'apel', 'rambutan'])`.

Perintah `split()`

Kebalikan dari `join()`, perintah `split()` digunakan untuk memecah string menjadi beberapa substring berdasarkan string tertentu. Hasil `split()` akan diperoleh list yang isinya tiap string. Contohnya:

```
buah = 'mangga dan apel dan durian'.split('dan')
print(buah)

buah = 'mangga apel durian'.split()
print(buah)
```

```
['mangga ', ' apel ', ' durian']
['mangga', 'apel', 'durian']
```

Perintah pertama pada contoh di atas, dimaksudkan untuk memecah string `'mangga dan apel dan durian'` berdasarkan string `'dan'`. Sehingga akan diperoleh list yang berisi 3 buah string, yaitu `'mangga '`, `' apel '`, dan `' durian'`.

Sedangkan dari contoh ke dua dapat kita lihat bahwa secara default perintah `split()` memecah string berdasarkan karakter spasi (*whitespace*).

Perintah `replace()`

Perintah replace() digunakan untuk mengubah substring dalam suatu string menjadi substring lainnya. Sebagai contoh misalkan terdapat string 'I LIKE PYTHON'. Selanjutnya substring 'LIKE' akan diganti 'LOVE' sehingga akan menjadi string baru 'I LOVE PYTHON'. Untuk melakukan hal ini dapat dilakukan menggunakan replace().

```
myStr = "I LIKE PYTHON"
myStr.replace("LIKE", "LOVE")
print(myStr)
```

```
I LIKE PYTHON
```

Berdasarkan contoh di atas, parameter pertama dari replace() merupakan substring yang akan diubah, dan parameter ke dua adalah substring yang baru.

Pengaturan Perataan Tampilan String

Di dalam Python, kita dapat dengan mudah untuk mengatur justifikasi posisi string ketika ditampilkan dengan perintah `print()`. Terdapat tiga perintah cara pengaturan justifikasi string yaitu: `rjust()`, `ljust()`, dan `center()`.

Perintah `rjust()`

Perintah `rjust()` digunakan untuk mengatur string dalam posisi rata kanan relatif terhadap *space* yang ditentukan. Contoh penggunaannya adalah:

```
buah = 'mangga'
print(buah.rjust(20))
print(buah.rjust(20, '.'))
```

```
              mangga
..............mangga
```

Berdasarkan contoh di atas, terdapat 20 *space* untuk mencetak string '`mangga`'. Pada contoh yang pertama, posisi string '`mangga`' berada di paling kanan dari *space* yang disediakan dan terdapat spasi kosong di depan string tersebut. Sedangkan pada contoh ke dua, di depan string terdapat karakter '.'.

Perintah `ljust()`

Kebalikan dari `rjust()`, perintah `ljust()` digunakan untuk perataan kiri dari string relatif terhadap *space* yang diberikan. Contohnya:

```
buah = 'mangga'
print(buah.ljust(20))
print(buah.ljust(20, '.'))
```

```
mangga
mangga..............
```

Perintah `center()`

Sedangkan perintah `center()` digunakan untuk memposisikan string di tengah-tengah *space* yang diberikan. Contoh:

```
buah = 'MENU'
print(buah.center(20))
print(buah.center(20,'*'))
```

```
        MENU
********MENU********
```