NTNU

Det skapende universitet

TDT4110 Informasjonsteknologi grunnkurs:

Tema: Mer om strenger

- 3rd edition: Kapittel 8

Professor Alf Inge Wang

Læringsmål og pensum

- Mål
  - Lære om
    - Grunnleggende operasjoner på strenger
    - Å skive/*slice* strenger
    - Teste, søke i og manipulere strenger
- Pensum
  - Starting out with Python:

  Chapter 8 More About Strings (3rd edition)

Tekststrenger (string)

- I Python kan en tekststreng sees på som en liste av tegn og bokstaver med fast lengde.
- En viktig forskjell på liste og tekststreng er at tekststreng ikke kan endre type eller deler av innhold.
- På samme måte som for lister, kan man få ut deler av strengen ved å bruke indeks og skiving (slice):

```
tekst = 'Dette er en test'
tekst[0] # Gir 'D'
tekst[14] # Gir 's'
```

Grunnleggende strengoperasjoner

Kapittel 8.1

Grunnleggende strengoperasjoner

- Mange slags programmer utfører operasjoner på strenger
- I Python finnes mange verktøy for å undersøke og manipulere strenger
  - Strenger er sekvenser, så mange av verktøyene som fungerer med andre sekvenser fungerer med strenger

Tilgang til enkelttegn i en streng

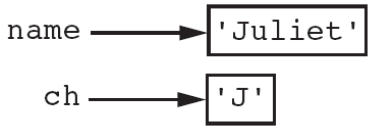
- For å få tilgang til de enkelte tegn i en streng
  - Bruk en for-løkke
    - Format: `for character in string:`
    - Nyttig når man skal iterere over en hel streng, eksempelvis for å telle antall forekomster av et visst tegn i den
  - Bruk indeksering
    - Hvert tegn har en indeks som spesifiserer dens plassering i strengen, og begynner med 0
    - Format: `character = my_string[i]`
- Lag funksjon `telle_tegn` som sjekker antall ganger et tegn befinner seg i en tekst.

kode: sjekk_streng.py

Iterasjon over strengen “Juliet”

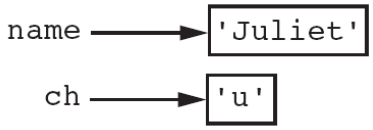
1st Iteration

```
for ch in name:
    print(ch)
```



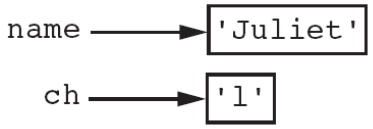
2nd Iteration

```
for ch in name:
    print(ch)
```



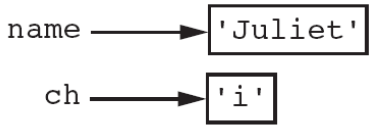
3rd Iteration

```
for ch in name:
    print(ch)
```



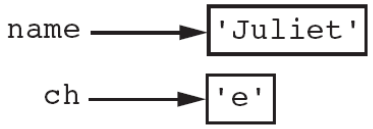
4th Iteration

```
for ch in name:
    print(ch)
```



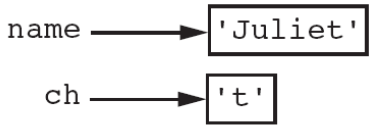
5th Iteration

```
for ch in name:
    print(ch)
```



6th Iteration

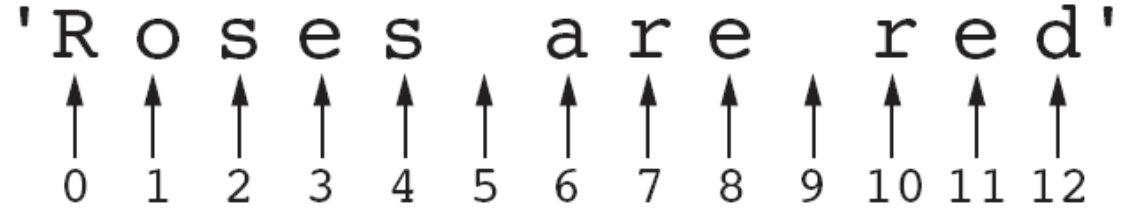
```
for ch in name:
    print(ch)
```



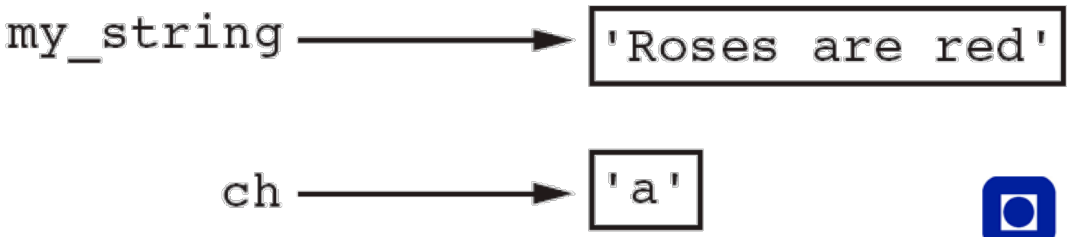
NTNU
Det skapende universitet

Tilgang til enkelttegn i en streng (forts.)

```
my_string = ‘Roses are red’
```




```
ch = my_string[6]
```



NTNU
Det skapende universitet

Tilgang til enkelttegn i en streng (forts.)

- IndexError-exception utløses hvis:
  - Du prøver å bruke en indeks som er utenfor strengens rekkevidde
    - Dette skjer nok hvis du forsøker å iterere videre når strengen slutter
- Funksjonen len(string) kan brukes til å finne strengens lengde på samme måte som for lister
  - Snedig for å hindre at man itererer forbi strengens slutt

Konkatinering av strenger

Konkatinering av strenger

- Konkatinering: å tilføye en streng ved slutten av en annen
  - Bruk operatoren + til å lage en streng som er en kombinasjon av dens operander
  - Den utvidede tilordningsoperatoren += kan også brukes til å konkatinere strenger
    - Operanden på venstre side av operatoren += må være en eksisterende variabel; i motsatt tilfelle utløses en exception

Tekststrenger og konkatenering (sammensetn)

Men man kan sette sammen flere tekststrenger ved å bruke + i mellom tekststrengene:

```
tekst = 'Dette '+ 'er '+'en '+'test'
# Gir 'Dette er en test'
```

Kan også sette sammen tekststrenger fra variabler:

```
nytekst = 'Her kommer: '+tekst
# Gir 'Her kommer: Dette er en test'
```

Hvis variabler med tall skal settes inn en streng, bruk str(variabel) som gjør om variabel til tekststreng:

```
tekst = 'Tallet er: '+str(tall)
```

Strenger er ikke-muterbare

- Når de først er laget, kan de ikke endres
  - Konkatinering endrer ikke faktisk noen av strengene, men lager tvert i mot en ny streng, og tilordner denne til en av de allerede eksisterende variablene
  - Man kan **ikke** bruke et uttrykk som dette:

```
string[index] = new_character
tekst[3] = ”Klare” # Gir feilmelding!!!
```

    - Dette vil gi en feilmelding

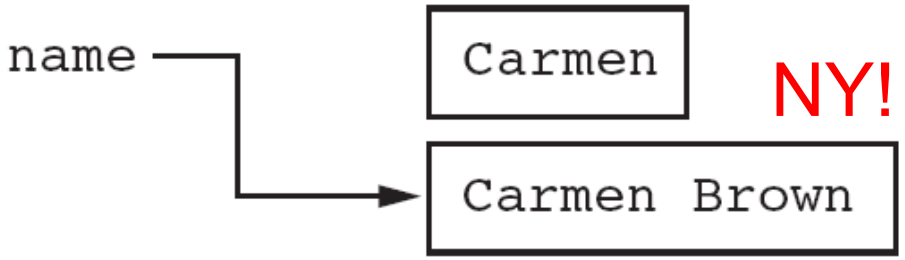
Strenger er ikke-muterbare (forts.)

- Tekststrengen `'Carmen'` blir tilordnet `name`:

```
name = 'Carmen'
```

- Tekststrengen `'Carmen Brown'` blir tilordnet `name`. Merk! Et nytt tekst objekt blir laget!!!

```
name = name + ' Brown'
```



NTNU
Det skapende universitet

Oppgave: finn tegn i tekst

- Skriv Python-koden til funksjonen `skriv_tegn_indeks`, som tar inn en parameterne `tekst` og `tegn`, og skriver ut alle indeksene der `tegn` befinner seg i strengen `tekst`.

kode: skriv_tegn_indeks.py

Slicing (skiving) av strenger

Kapittel 8.2

Slicing av strenger (som for lister)

- Slice: spennvidde av enheter tatt fra en strengsekvens, kjent som en *substring*
  - Format: `string[start : end : steg]`
    - Uttrykket vil returnere en streng som har en kopi av tegnene fra start til, men ikke medregnet end
    - Hvis start er uspesifisert, antas indeks 0
    - Hvis end er uspesifisert, antas indeks len(string)

```
tekst = 'Dette er en test'
tekst[0:3] # Gir 'Det'. Samme som tekst[:3]
tekst[12:16] # Gir 'test'. Samme som tekst[12:]
tekst[::2] # Gir 'Dtee nts'. (annenhvert tegn)
```

Testing, søking og manipulering av strenger

Kapittel 8.3

NTNU
Det skapende universitet

Testing, søking og manipulering av strenger

- Du kan bruke operatoren in til å avgjøre om en streng inneholdes av en annen streng (samme som lister)
  - Generelt format: `streng1 in streng2`
    - streng1 og streng2 kan være *string literals* eller variable som refererer til strenger
  - Tilsvarende kan du bruke operatoren not in til å avgjøre om en streng ikke er inneholdt av en annen streng

Strengmetoder

- Strenger i Python har mange metoder, inndelt på typer av operasjoner
  - Generelt format:
    - `mystring.method(arguments)`
  - Noen metoder tester en streng for spesifikke karaktertrekk
    - Disse metodene er såkalte boolske metoder som returnerer enten `True` hvis betingelsen er sann ellers `False`.

Noen strengteste-metoder:

Method	Description
`isalnum()`	Returns true if the string contains only alphabetic letters or digits and is at least one character in length. Returns false otherwise.
`isalpha()`	Returns true if the string contains only alphabetic letters, and is at least one character in length. Returns false otherwise.
`isdigit()`	Returns true if the string contains only numeric digits and is at least one character in length. Returns false otherwise.
`islower()`	Returns true if all of the alphabetic letters in the string are lowercase, and the string contains at least one alphabetic letter. Returns false otherwise.
`isspace()`	Returns true if the string contains only whitespace characters, and is at least one character in length. Returns false otherwise. (Whitespace characters are spaces, newlines (`\n`), and tabs (`\t`).
`isupper()`	Returns true if all of the alphabetic letters in the string are uppercase, and the string contains at least one alphabetic letter. Returns false otherwise.

Strengmetoder (forts.)

- Noen metoder returnerer en kopi av en streng som det er gjort forandringer på
  - De ”simulerer” at strenger er muterbare
- Sammenlikning av strenger foregår case-sensitivt
  - Store bokstaver skilles fra små bokstaver
  - lower og upper kan brukes til å gjøre sammenlikning av strenger uten å ta hensyn til *case*

Strengmodifiserings-metoder:

Method	Description
`lower()`	Returns a copy of the string with all alphabetic letters converted to lowercase. Any character that is already lowercase, or is not an alphabetic letter, is unchanged.
`lstrip()`	Returns a copy of the string with all leading whitespace characters removed. Leading whitespace characters are spaces, newlines (`\n`), and tabs (`\t`) that appear at the beginning of the string.
`lstrip(`*`char`*`)`	The *`char`* argument is a string containing a character. Returns a copy of the string with all instances of *`char`* that appear at the beginning of the string removed.
`rstrip()`	Returns a copy of the string with all trailing whitespace characters removed. Trailing whitespace characters are spaces, newlines (`\n`), and tabs (`\t`) that appear at the end of the string.
`rstrip(`*`char`*`)`	The *`char`* argument is a string containing a character. The method returns a copy of the string with all instances of *`char`* that appear at the end of the string removed.
`strip()`	Returns a copy of the string with all leading and trailing whitespace characters removed.
`strip(`*`char`*`)`	Returns a copy of the string with all instances of *`char`* that appear at the beginning and the end of the string removed.
`upper()`	Returns a copy of the string with all alphabetic letters converted to uppercase. Any character that is already uppercase, or is not an alphabetic letter, is unchanged.

kode: strenger2.py

Sammenlikningsmetoder

- Programmer trenger ofte å søke etter substrenger
- Flere metoder finnes for å gjøre dette:
  - `endswith(substring)`: sjekker om en streng slutter med substring
    - Returnerer `True` eller `False`
  - `startswith(substring)`: sjekker om en streng begynner med substring
    - Returnerer `True` eller `False`

Søk- og erstatt-metoder

- Flere metoder finnes (forts.)
- `find(substring)`: leter etter substring i strengen
  - Returnerer den laveste indeksen av substrengen, eller -1 om substrengen ikke finnes i strengen
- `replace(substring,new_string)`:
  - Returnerer en kopi av strengen hvor alle forekomster av substring er erstattet med new_string

Oversikt over søk- og erstattmetoder

Method	Description
`endswith(substring)`	The *substring* argument is a string. The method returns true if the string ends with *substring*.
`find(substring)`	The *substring* argument is a string. The method returns the lowest index in the string where *substring* is found. If *substring* is not found, the method returns −1.
`replace(old, new)`	The *old* and *new* arguments are both strings. The method returns a copy of the string with all instances of *old* replaced by *new*.
`startswith(substring)`	The *substring* argument is a string. The method returns true if the string starts with *substring*.

Å splitte opp en streng

- Metoden `split`: returnerer en liste som inneholder ordene i strengen
  - Bruker space (mellomrom) som skilletegn som standard
  - Kan bruke et annet skilletegn gjennom å sende det som argument til metoden split
  - Eks:

```
tekst = 'Dette er en test'
print(tekst.split())
```

Gir:

```
['Dette', 'er', 'en', 'test']
```

Oppgave: overskrift

- Lag funksjonen `overskrift`, som tar inn en parameteren `tekst` og returnerer samme tekststreng der alle ordene starter med stor forbokstav.
  - Benytt: split(), upper(), for-løkke

kode: overskrift.py

Oppsummering

Dette kapittelet dekket:

- Operasjoner på strenger, som
  - Metoder for å iterere gjennom strenger
  - Operatorer for repetisjon og konkatinering
  - Strenger som ikke-muterbare objekter
  - Å *slice* og teste strenger
  - Metoder for strenger
  - Å splitte opp strenger