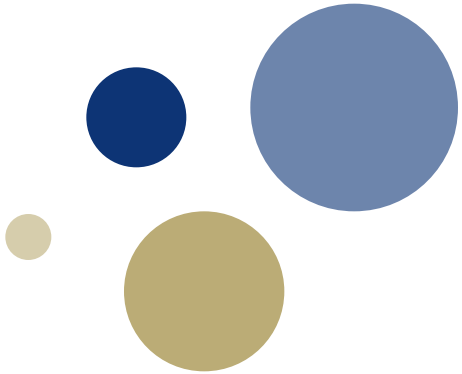
NTNU

Kunnskap for en bedre verden



Python: Strenger

3. utgave: Kapittel 8

TDT4110 IT Grunnkurs

Professor Guttorm Sindre

Læringsmål og pensum

- Mål
  - Forstå
    - Hva strenger er
    - Grunnleggende operasjoner på strenger
    - Indeksering av tegn i strenger, inkl. *slicing*
  - Kjenne til noen vanlige funksjoner og metoder på strenger
  - Kunne løse praktiske programmeringsproblemer med strenger

- Pensum
  - Starting out with Python:

  Chapter 8 More About Strings (3rd edition)

Tekststrenger (string)

- HVORFOR trenger vi dette?
  - Menneskelig informasjon: tekst spiller en sentral rolle
  - Veldig mange nyttige programmer behandler tekst
    - Tekstbehandling, weblesere
    - Journalistikk, publisering
    - Saksbehandlingssystemer
    - Teknisk informasjon, brukerveiledninger, …
- I Python kan en tekststreng sees på som en liste av tegn og bokstaver med fast lengde
  - Men i motsetning til liste er den ikke muterbar

Grunnleggende strengoperasjoner

Kapittel 8.1

Grunnleggende strengoperasjoner

- I Python finnes mange funksjoner / metoder for å undersøke og manipulere strenger
  - Strenger er sekvenser,
  - Mange funksjoner/metoder tilsvarer de for lister og tupler
- 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’
tekst[-1]       # Gir ’t’, som: tekst[len(tekst)-1]
```

Oppgave: loddrett tekst

- **Lag en funksjon som får inn en tekststreng som parameter og skriver ut teksten loddrett**
  - **Ekstra hvis fort ferdig: skriv også ut teksten baklengs**

- Hint: For å få tilgang til enkelttegn i en streng
  - For-løkke: **for character in string:**
  - Indeksering av enkelttegn:
    - På samme måte som enkeltelementer i lister
    - Indeks begynner med 0, viser tegnets plassering i strengen
    - Format: **character = my_string[i]**
    - For å finne en strengs lengde: **len(string)**
    - (IndexError: du har brukt en indeks utenfor strengen)

Konkatenering av strenger

- Konkatenering: sette sammen to strenger til en
  - Bruk operatoren + , f.eks. navn = fornavn + etternavn
  - evt. += hvis du har en variabel hvor du samler opp tekst
    - F.eks. setning += nytt_ord
  - Hvis du skal sette et tall inn i en streng…
    - str( )-funksjonen konverterer tall til tekst
    - F.eks. respons = 'Du har ' + str(saldo) + 'kroner på kontoen'

Strenger er ikke-muterbare

- Når de først er laget, kan de ikke endres
  - Samme som for tupler (mens lister er muterbare)
- Uttrykk som kan se ut som endring
  - lager egentlig en ny streng
  - tilordner variabelen til denne i stedet
    - setning += nytt_ord
    - navn = navn.replace('å','aa') #bytter ut alle å med aa
    - setning = setning.upper( ) #alt blir store bokstaver
  - Forsøk på å endre tegn inni en streng vil gi feilmelding:
    - setning[i] = 'w'
    - tekst[3] = "Klare"

Slicing (skiving) av strenger

Kapittel 8.2 / 9.2

Slicing av strenger (a la lister)

- Slice: en substreng tatt fra en streng
  - Samme prinsipp og syntaks som slicing av lister
  - Format: string[start : end : steg]
    - kopi av tegnene fra start til, men ikke med, end
    - Hvis start er uspesifisert, antas indeks 0
    - Hvis end er uspesifisert, antas indeks len(string)
    - Hvis steg er uspesifisert (mest vanlig), antas 1

```
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 / 9.3

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
    - Kan f.eks brukes som betingelse i if- og while-setninger
  - Fins også en motsatt operator not in

- streng.index(tegn) – finne hvor i en streng et tegn er

Strengmetoder

- Strenger i Python har mange ferdige metoder
  - Generelt format: mystring.method(arguments)
  - Testemetoder
    - Betingelser for hele strenger
    - Returnerer True hvis betingelsen er sann ellers False.
  - Modifiseringsmetoder:
    - Kopier av strengene hvor noe kan være endret
  - Søk- og erstatt-metoder
    - Leter etter tegn / delstrenger i strenger

Testemetoder:

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.

Modifiseringsmetoder:

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.

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 ei liste som inneholder ordene i strengen
  - Bruker space (mellomrom) som default skilletegn
  - Kan gi inn annet skilletegn som argument
  - Eks:

```
tekst = ’Dette er en test’
print(tekst.split())       # Gir: ['Dette', 'er', 'en', 'test']
print(tekst.split(’e’))    # Gir: [’D’, ’tt’, ’ ’, ’r ’, ’n t’, ’st’]
```

Eksempel: omformattering av navn

Inn: Fnavn (Mnavn) Enavn, returner Enavn, Fnavn (Mnavn)

Med slicing:

```
def enavn_fnavn(navn):
    i = navn.rindex(' ')
    fornavn=navn[:i]
    etternavn=navn[i+1:]
    return (etternavn + ', ' +
                fornavn)
```

Med split( )

```
def enavn_fnavn(navn):
    liste=navn.split( )
    siste=len(liste)-1
    etternavn=liste[siste]
    fornavn=liste[0]
    for i in range(1,siste-1):
        fornavn+=liste[i]
    return (etternavn + ', ' +
                fornavn)
```

Oppgave: omgjøre navn

- Lag en funksjon fnavn_enavn(navn)
  - Inn: ’Etternavn, Fornavn Evt Mellomnavn’
  - Returner: ’Fornavn Evt Mellomnavn Etternavn’
  - Velg selv om du vil bruke slice eller split

- Ekstraoppgave hvis raskt ferdig:
  - lignende funksjon, men lag initial(er) i stedet for mellomnavn
  - Inn: ’Enavn, Fnavn Evt Mellom’, Retur: ’Fnavn E.M. Enavn’

kode: navn_V0.py

løsn: navn_V1.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