IN1140, H2021 – gruppetime oppgaver
Introduksjon til Tekst i Python

I disse oppgavene skal vi introdusere Python, og vise hvordan vi kan jobbe med tekst i Python. Nærmere bestemt kommer vi her til å jobbe med strenger, lister, hvordan vi kan printe verdier, lese inn filer, og jobbe med tekststrenger i Python. Dersom du ønsker å jobbe videre med Python på egenhånd anbefaler vi at dere jobber med Trix-oppgavene for IN1140 (se emnesiden).

1 Hva er Python?

Python er et programmeringsspråk. Dere skal lære å programmere med Python i dybde i IN1000. I IN1140, skal vi i hovedsak fokusere på bruk av Python for tekstdata.

Python har et eget vokabular, eller mengde med reserverte ord, som har en spesiell betydning i Python og er svært viktige. Når Python møter slike ord i et program/kode har ordet en og bare en betydning for Python. Noen av disse ordene er: `and`, `break`, `class`, `continue`, `def`, `elif`, `else`, `for`, `from`, `if`, `import`, `in`, `not`, `or`, `while`.

Når du skal skrive et program skal du kunne bruke dine egne ord som har betydning for deg og koden din. Disse ordene kalles variabler. Dine variabelnavn kan være alle ordene i vårt menneskelig vokabular, men du **må aldri** bruke et av Python sine reserverte ord som variabelnavn (ellers kommer Python til å tro at du mener noe annet enn det du faktisk mener, siden reserverte ord kun har én betydning for Python).

1.1 Installasjon

Om du ikke allerede har installert Python på maskinen din, vil gruppelæreren hjelpe deg med det.

2 Strenger i Python

En streng er en sekvens av symboler som også kan inneholde mellomrom og siffer. For eksempel både ordet “sitron” og frasen “kurven inneholder 3 sitroner” er strenger. En streng kan også være en sekvens av tall, feks “123456”. For at Python skal kunne gjenkjenne at en variabel er av datatypen streng, må du alltid ha strengverdien mellom anførselstegn.

Eksempler på egendefinerte variabler av datatypen streng:

```
>>> minstreng = "Hello world!"
>>> nystreng = "123456"
>>> sistestreng = "Jeg sto opp kl.5.00 i dag morges."
```

Du kan be Python om å vise deg verdien av en variabel ved å printe den på skjermen. Dette gjøres ved bruk av den innebygde funksjonen “`print()`”. For eksempel, for å vise verdien av variabelen *minstreng*, må du skrive:

```
>>> print(minstreng)
```

Da får du som resultat:

```
>>> print(minstreng)
Hello world!
```

Om du vil printe strengen "Hello world!" uten å kalle variabelen *minstreng* kan du skrive:

```
>>> print("Hello world!")
Hello world!
```

2.1 Strengmetoder

Python Library Reference inneholder en oversikt over metoder som er tilgjengelig for forskjellige datatyper. For strenger, finner du relevant informasjon på: `http://docs.python.org/library/stdtypes.html`. Alternativt kan du skrive inn `help(str)` i et Python-shell for å få en oversikt over metodene som er tilgjengelig for strenger. Observer at det er flere metoder enn de som vises på skjermen og bruk "space" for å navigere nedover. Bruk "q" for å avslutte.

Vi skal se på et par metoder som er tilgjengelige for strenger:

- **`count()`**:

```
|  count(...)
|      S.count(sub[, start[, end]]) -> int
|
|      Return the number of non-overlapping occurrences of
|      substring sub in
|      string S[start:end].  Optional arguments start and end
|      are interpreted
|      as in slice notation.
```

Dette betyr at du for eksempel kan bruke `count` slik :

```
>>> mystring = "hello"
>>> print(mystring.count("l"))
2
```

- **split()**:

```
|   split(...)
|      S.split(sep=None, maxsplit=-1) -> list of strings
|
|      Return a list of the words in S, using sep as the
|      delimiter string.  If maxsplit is given, at most maxsplit
|      splits are done. If sep is not specified or is None, any
|      whitespace string is a separator and empty strings are
|      removed from the result.
```

Dette betyr at du for eksempel kan bruke `split` slik :

```
>>> mystring = "Hello world!"
>>> mystr.split(" ")
["hello", "world!"]
>>> mystring.split("l")
["he", "", "o wor", "d!"]
>>> mystr.split("o")
["hell", " w", "rld!"]
>>> mystr.split("!")
["hello world", ""]
```

- **endswith()**:

```
|  endswith(...)
|      S.endswith(suffix[, start[, end]]) -> bool
|
|      Return True if S ends with the specified suffix, False otherwise.
|      With optional start, test S beginning at that position.
|      With optional end, stop comparing S at that position.
|      suffix can also be a tuple of strings to try.
```

Dette betyr at du for eksempel kan bruke `endswith` slik :

```
>>> mystring = "Hello world!"
>>> mystring.endswith("!")
True
>>> mystring.endswith("d")
False
>>>
```

- **lower()**:

```
|  lower(...)
|      S.lower() -> str
|
|      Return a copy of the string S converted to lowercase.
```

Dette betyr at du for eksempel kan bruke `lower` slik :

```
>>> mystring = "Hello world!"
>>> mystring.lower()
"hello world!"
>>> mystring = "HELLO WORLD!"
>>> mystring.lower()
"hello world!"
```

3 Lister i Python

Lister i Python inneholder en samling elementer. Elementene i en liste kan være ord, tall, fraser, setninger, osv. I Python blir lister skrevet med firkantede parenteser og elementene er skilt av et komma:

```
["dette", "er", "en", "liste"]
```

.

Du kan opprette en tom liste ved å skrive:

```
>>> mylist = []
```

3.1 Liste metoder

- Du kan legge til elementer til listen din ved å bruke `append()` metoden. Elementene vil alltid bli lagt til på slutten av listen. Du kan legge til samme elementet flere ganger.

```
|  append(...)
|      L.append(object) -> None -- append object to end
|
```

For eksempel:

```
>>> mylist = []
>>> mylist.append("første")
>>> print(mylist)
["første"]
>>> mylist.append("første")
>>> print(mylist)
["første","første"]
>>> mylist.append("andre")
>>> print(mylist)
["første","første", "andre"]
```

- Du kan telle forekomster av en viss verdi ved å bruke list metoden `count()`:

```
|  count(...)
|      L.count(value) -> integer -- return number of occurrences of value
|
```

For eksempel:

```
>>> mylist.count("første")
2
>>> mylist.count("andre")
1
```

- Du kan slette den første forekomsten av en verdi ved å bruke metoden `remove()`[1]:

```
|  remove(...)
|      L.remove(value) -> None -- remove first occurrence of value.
|      Raises ValueError if the value is not present.
|
```

```
>>> mylist.remove("første")
>>> mylist
["første", "andre"]
```

- Du kan sortere listen din (alfabetisk og ikke-alfabetisk rekkefølge) ved å bruke metoden `sort()`:

```
|  sort(...)
|      L.sort(key=None, reverse=False) -> None -- stable sort *IN PLACE*
```

```
>>> mylist.sort()
>>> mylist
["andre", "første"]
>>> mylist.sort(reverse=True)
>>> mylist
["første", "andre"]
```

I tillegg til disse metodene kan sekvenstypene i Python (strenger og lister) manipuleres med slice-notasjon. Denne notasjonen lar oss lett hente ut en del av en større sekvens. Slik at [x:y] betyr fra og med x, til (men ikke inkludert) y, og begynner alltid å telle fra 0. For eksempel:

```
>>> liste = ["a", "b", "c", "d"]
>>> liste[2:4]
["c", "d"]
>>> liste[:-1]
["a", "b", "c"]
>>> streng = "abcde"
>>> streng[2:4]
"cd"
>>> streng[:-1]
"abcd"
```

4 Åpne og lese filer i Python

Her skal dere bli tildelt to filer av gruppelæreren: *minførstePython.py* og *test.txt*.

- *test.txt* inneholder en kort tekst hentet fra `https://www.nrk.no`[2].
- *minførstePython.py* er en tom fil som du skal fylle ut med litt Python kode.

Python er i utgangspunktet ikke så glad i norske tegn. Vi må derfor ha en spesiell kommentar som første linje i *minførstePython.py* filen for å få dette til å virke:

```
# encoding: utf-8
```

[1] MERK: når du jobber på kommandolinjen, kan du skrive variabelnavnet direkte for å printe dens verdi, som vist i eksempelet under. Hvis du skriver et program i en ".py" fil, må du alltid bruke `print()`

[2] `https://www.nrk.no/troms/holdt-pa-a-gi-opp-boligjakten-_-na-blir-et-hotellrom-hjemmet-hennes-1.14168915`

For å lese inn en fil som en streng må du åpne den, lese den og lukke den:

```
f = open("<sti-til-filen>")
filinnhold = f.read()
f.close()
```

For å åpne filen *test.txt* må du derfor gi stien til filen i maskinen din. Gruppelæreren kan hjelpe deg med det om du ikke er sikker på hvordan det skal gjøres.

```
>>> f = open("test.txt")
>>> filinnhold = f.read()
>>> filinnhold
'Da Natalie Netland hadde svart på 40 boligannonser uten å få ett eneste napp, var
hun i ferd med å gi opp drømmen om å bli barnehagelærer. Så fikk hun et uventa
tilbud om både kost og losji.\\n"Nå kan jeg bo her til jeg får flytte for meg
selv" sier Egersund-jenta takknemlig der hun står utenfor ett av Tromsøs hoteller.
\\n...'
>>> f.close()
```

Ser du tegnene "\n"? Dette tegnet betyr "new line" og er maskinen sin måte å forstå at en linjeskifte har blitt brukt.

En annen måte å lese inn en fil i Python på er å lese inn filen *test.txt* som en liste, der elementene i listen tilsvarer en linje i filen. Dette gjør vi med en såkalt `for`-løkke. En `for` løkke er en måte å få Python til å iterere over elementer i en fil (eller liste/streng). Den brukes slik:

```
f = open("<sti-til-filen>")
for line in f:
    ...
f.close()
```

For eksempel:

```
>>> f = open("test.txt")
>>> for line in f:
    print(line)
...
Da Natalie Netland hadde svart på 40 boligannonser uten å få ett eneste napp, var
hun i ferd med å gi opp drømmen om å bli barnehagelærer. Så fikk hun et uventa
tilbud om både kost og losji.

"Nå kan jeg bo her til jeg får flytte for meg selv" sier Egersund-jenta takknemlig
der hun står utenfor ett av Tromsøs hoteller.


.
.
.
>>>
>>> f.close()
```