Läsning och utskrift, formattering

Användning backslashtecknet dvs tecknet \ vid utskrifter

Tecknet enkelblipp `'` används för att avgränsa den sträng som skall skrivas ut, exvis: `print('Hej')`

Om den text som skall skrivas ut skall innehålla enkelblipp kan det lösas genom att använda backslashtecknet, dvs tecknet `\`.

Exempel: Om man önskar skriva ut texten `XX''YY` kan det göras med:

```
print('XX\'\'YY')
```

Göra en tab-utskrift görs med `\t`: ex `print('A\tB\tC')` ger utskriften:

```
A    B    C
```

Radbyte görs med `\n`, ex `print('AB\nC')` ger utskrift av AB följt av C med nästa rad.

Utskrift av backslash-tecknet, ex: `print('\\\\')` ger utskriften: `\\`

Läsning och utskrift, formattering

```
namn = input('Vad heter du? ')
print('namn')
print(namn)
print('Ditt namn = ', namn)
```

```
Vad heter du? Kim
namn
Kim
Ditt namn = Kim
```

input-satsen (funktionen input) väntar på att användaren skriver något på tangentbordet och avslutar med return/enter. Det som användaren knappar in tilldelas variabeln `namn`, som automatiskt blir en sträng, dvs typen är *str*.

Printsatsen består av ett antal *parametrar* som separeras med kommatecken. Parametrarna kan vara strängar och/eller variabler. Kommatecknet ger ett blanktecken

Den inbyggda funktionen *input* ger alltid som resultat en sträng.

```
namn = input('Vad heter du? ')
print('Ditt namn = ', namn)
print('typ av namn', type(namn))
```

```
Vad heter du? Kim Skog
Ditt namn =  Kim Skog
typ av namn <class 'str'>
```

```
namn = input('Vad heter du? ')
print('Ditt namn = ', namn)
print('typ av namn', type(namn))
```

```
Vad heter du? 1234
Ditt namn =  1234
typ av namn <class 'str'>
```

Vad händer om man matar in return/enter direkt?

```
namn = input('Vad heter du? ')
print('Ditt namn = ', namn)
print('typ av namn', type(namn))
if namn == '': #namn en tom sträng?
    print('En tom sträng')
```

```
Vad heter du?
Ditt namn =
typ av namn <class 'str'>
En tom sträng
```

Om vi önskar läsa in numeriska data måste strängen konverteras.

```
text = input('Ge ett heltal: ')
tal = int(text)
print('Talet = ', tal)
```

```
Ge ett heltal: 14
Talet =  14
```

```
text = input('Ge ett flyttal: ')
tal = float(text)
print('Talet = ', tal)
```

```
Ge ett flyttal: -3.7
Talet =  -3.7
```

Funktionen *int* konverterar till ett heltal
Funktionen *float* konverterar till ett flyttal

Läsa in många tal:

```
x = input('Ge ett antal heltal: ')
print('x=', x)
print('type(x)=', type(x))
```

```
Ge antal heltal: 7 88 9
x= 7 88 9
type(x)= <class 'str'>
```

```
y = x.split()
print('y=', y)
print('type(y)=', type(y))
```

```
y= ['7', '88', '9']
type(y)= <class 'list'>
```

För att konvertera listan med strängar till heltal måste vi ha en loop som konverterar varje sträng

```
z = [int(e) for e in y]
print('z=', z)
print('type(z)=', type(z))
```

```
z= [7, 88, 9]
type(z)= <class 'list'>
```

Kursivt: F-strings (formatted string literals)

Används för att redigera utskrifter/strängar och få dem att bli lite snyggare.

```
x = 1.5
y = x*x
print('y är', y)
```

```
y är 2.25
```

```
x = 1.6
y = x*x
print('y är', y)
```

```
y är 2.5600000000000005
```

Inte så snyggt. Det verkar vara som de första 15 decimala sifrrorna är korrekta. Det beror på att tal lagras binärt. 2.56 kan uttryckas binärt:

$1*2^1 + 0*2^0 + 1*2^{-1} + e = 2 + 0.5 + e$, där $e = 0.06$.

Men vad är e uttryckt binärt? Jo, $1/32 + e = 1*2^{-6} + e = 0.03125 + e$, där $e = 0.02875$. Men vad är detta binärt? Jo ...

Med f-strings

```
print(f'y är {y:.2f}')
```

```
y är 2.56
```

- f i början anger att det är en f-string, en sträng skall byggas
- {…} kallas *platshållare*
- Att det står `y` i platsbehållaren innebär att variabeln `y` skall skrivas ut.
- Att det dessutom står `:.2f` i platsbehållaren innebär att utskriften av `y` skall *formatteras* så att det skrivs som en *float* (f) ut med två decimaler och avrundas till detta.

```
print(f'y är {y:8.2f}')
```

```
y är     2.56
```

```
print(f'y är {y:.2e}')
```

```
y är 2.56e+00
```

```
print(f'y är {(10*y):.3e}')
```

```
y är 2.560e+01
```

```
name = 'Saga'
age = 24
print(f'Hello, {name}. You are {age}.') # 2 st platshållare
```

```
Hello, Saga. You are 24.
```

```
print(f'Hello, {name:8}. You are {age:6}.')
```

```
Hello, Saga    . You are     24.
```

f-print kan användas till att bygga strängar:

```
# Skapa en sträng mha f-string
s = f'Hello, {name:8}. You are {age:6}.'
print(s)
```

```
Hello, Saga    . You are     24.
```

Kursivt: **Format-metoden**

format är en metod som kan användas på strängar

```
txt = 'Hello, {name}. You are {age}.' # 2 st platshållare
print(txt.format(name = 'Saga', age = 24))
```

```
Hello, Saga. You are 24.
```

```
txt = 'Hello {0:8}. You are {1:6}'.format(name,age)
print(txt)
```

```
Hello, Saga    . You are     24.
```

```
txt ='Hello {}. You are {}'.format('John',36)
print(txt)
```

```
Hello John. You are 36
```