Output-fördjupning

Programmering 1

ht 20

Input-output

Input

`Input` är all information datorn får utifrån. Det kan tillexempel vara från tangentbordet eller musen. Vi har än så länge bara tittat på funktionen `input` som stannar programmet och väntar på användar-input från tangentbordet.
`Input` går att använda på två sätt:

1. `input()` — Pausar programmet och väntar på att användaren trycker `enter`.
2. `input("Text")` — Skriver ut `text`, pausar programmet och väntar på att användaren trycker `enter`

Input-output

Output

`Output` är den information datorn skickar ut. Än så länge har det varit i form av text i konsolen och i form av färg i Turtle. För att göra en `output` använder man `print()`. Innanför paranteserna markerar man text som ska skrivas med antingen citat-tecken eller appostrofer. Det viktiga är att det börjar och slutar på samma. Man kan även skriva ut värdet på variabler.

```
1  >>> print("Vilket vackert väder")
2  Vilket vackert väder
3  >>> a = 103
4  >>> print(a)
5  103
```

Input-output

Output

Vi kan också skriva ut flera saker i samma `print`:

```
1 >>> a = 5
2 >>> print(a, "detta är text", a, a+a, 'mer text'
      , 'ny text')
3 5 detta är text 5 10 mer text ny  text
```

Efter ett komma blir det ett mellanslag. Vill man undvika det får man konvertera delarna till strängar och slå ihop dem.

```
1 >>> print(str(a)+"detta är text"+'mer text')
2 5detta är textmer text
```

Input-output

\n

Vi har pratat om att man kan tvinga fram en radbrytning i en print med hjälp av **\n**

```
1 >>> print("En hel\nmening")
2 En hel
3 mening
```

Input-output

End

Efter varje `print()` så gör Python en radbrytning. Detta är för att det finns ett dolt kommando man kan skicka med till `print()`. I funktionen `print()` finns en variabel, `end`, som innehåller vad som händer när `print`-funktionen har skrivit ut allt.

```
1 print("Detta är en mening.", end=" ")
2 print("Detta är en ny mening.")
3 print("Fin.")
```

```
1 Detta är en mening. Detta är en ny mening.
2 Fin.
```

I varje `print`-anrop så återställs variabeln `end`.

Input-output

Seperator

Vill man förändra vad som händer mellan varje del som skrivs ut så får man ändra på den inbyggda variabeln **sep** i **print**-funktionen. Som standard är **sep=" "**, alltså ett mellanslag. Det är därför det blir mellanslag mellan variabler.

```
1 print("Ett ord", "är", "lite.", sep="5")
2 print("Hej", "då", "re", sep="\n")
```

```
1 Ett ord5är5lite.
2 Hej
3 då
4 re
```

Input-output

Tillsammans

Man kan använda båda i en och samma `print`-anrop.

```
1 >>> print("Ett litet","slott", sep=":-:", end="
      !! w00p !!")
2 Ett litet:-:slott!! w00p !!
```

Input-output

String interpolation

```
>>> a = "Hej"
>>> b = "Nej"
>>> c =  3
>>> print("Lite text", a, "som är helt", b, "
    Nonsensig", c, "!!")
Lite text Hej som är helt Nej Nonsensig 3 !!
```

Sedan Python 3.6 kan man skriva så här istället:

```
>>> a = "Hej"
>>> b = "Nej"
>>> c =  3
>>> print(f"Lite text {a} som är helt {b}
    Nonsensig {c} !!")
Lite text Hej som är helt Nej Nonsensig 3 !!
```

Notera `f`:et som är före strängen.

Input-output

Tab

Tabbar formaterar saker vackert i utskrifter:

```
1 print("1 \t 2 \t 3")
2 print("Ord \t är \t fina")
```

```
1 1    2    3
2 Ord  är   fina
```

Övningar

Input-Output

1. Skriv ett program som tar emot ett namn och hälsar med String interpolation
2. Skriv ett program som tar emot ett par ”ord” och skriver ut dem på en rad med :) mellan varje ”ord”.
3. Skriv ett program som skriver ut en rad med en *, sen två *, sen tre och upp till tolv stjärnor

```
1 *
2 **
3 ***
```

4. Centrera stjärnorna (det är okej att ändra till 1,3,5 stjärnor av estetiska skäl):

```
1       *
2      ***
3     *****
```