Variabler och datatyper

Stefan Holmberg, Systementor AB

Vi smakar på olika Datatyper + expressions

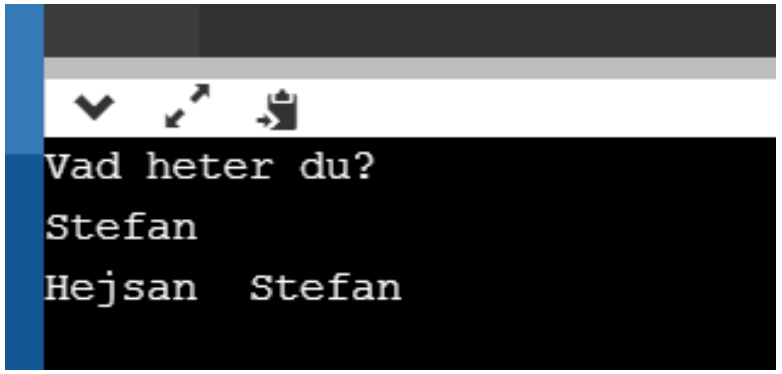
Är det skillnad på

print(”2019-10-31”) # med fnuttar

print(2019-10-31) # utan fnuttar

Vi smakar på inmatning och variabler

Programmet ska fråga vad du heter. Sedan skriva ut
Hejsan <det du matade in>



VI KODAR TILLSAMMANS

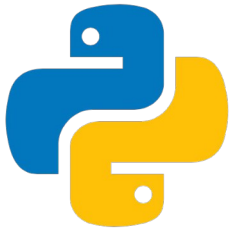
En del nya begrepp (input???) blev det...lugn...det kommer!

Python: variabler

- Variabler finns i ALLA språk
- Tänk på det som en LÅDA där du kan lägga nåt
  - Lådan har ett namn (som du bestämmer).
  - För att använda det som ligger i lådan för utskrift/beräkning osv så kan vi få dess värde genom lådans namn

- Som du alltså plockar fram sen när det behövs
  - Mellanlagring, delberäkning. ”ge mig värdet i lådan med namnet ’Age’”

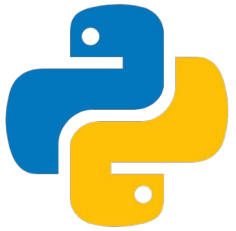
Lab Din första Variabel



Få detta att funka och vad blir resultatet?

```
antalDagarPåKursen = 25
print("Så här många dagar är kursen:", antalDagarPåKursen)
```

Så... vad har vi vunnit???



Variabel = varierande värde över tid – lägg till två rader till

```
antalDagarPåKursen = 25
print("Så här många dagar är kursen:", antalDagarPåKursen)

antalDagarPåKursen = antalDagarPåKursen + 50
print("Men den är så rolig så den borde ju vara:", antalDagarPåKursen)
```

Python: variabler

- C:
  - Deklarera variabler med t.ex:
  - `int x;`
  - Sätta variabelns värde med t.ex:
  - `x=42;`

- Python:
  - Variabler behöver inte deklareras i förväg som med C
  - Variabelns typ räknas ut implicit utifrån vad du sätter den till:
  - `x=42   # x är implicit en int`
  - `x=”Hello” # nu är x en string`

Variabler

- Mellanlagring
- “jag har inte på mig ALLA mina strumpor på en gång. Jag stoppar undan tills jag behöver”

DATATYPER:

Int = en låda där man kan stoppa HELTAL
Float = en låda där man kan stoppa FLYTTAL (decimaltal)
String = en låda där man kan stoppa en TEXT
Bool = en låda där man kan stoppa ett av två värden, TRUE eller FALSE

Python: variabler

- Python håller koll på typer åt dig och hindrar fel
- T.ex:

```
x=42
y="Hello"
z=x+y
print(z)
```

```
Traceback (most recent call last):  File "main.py", line 4,
  in <module>
    z=x+y
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

TESTA SJÄLV

```
x="I say "
y="Hello"
z=x+y
print(z)
```

REFLEKTION: **När** blir det fel??? Varför?

Python: variabler

- print() kan hantera olika typer:

```
a=42
b=42.0
c=float(42)


d="42"

print(a)
print(b)
print(c)
print(d)     #testa d = d + ”12”
```

Operators

Som i matematiken = parenteser, ordning etc spelar roll

```
print(2 * 3 % 5)
```

```
print((5 * ((25 % 13) + 100) / (2 * 13)) // 2)
```

Operators	Meaning
`()`	Parentheses
`**`	Exponent
`+x`, `-x`, `~x`	Unary plus, Unary minus, Bitwise NOT
`*`, `/`, `//`, `%`	Multiplication, Division, Floor division, Modulus
`+`, `-`	Addition, Subtraction
`<<`, `>>`	Bitwise shift operators
`&`	Bitwise AND
`^`	Bitwise XOR
`|`	Bitwise OR
`==`, `!=`, `>`, `>=`, `<`, `<=`, `is`, `is not`, `in`, `not in`	Comparisons, Identity, Membership operators
`not`	Logical NOT
`and`	Logical AND
`or`	Logical OR

More on input/output

- input = returnerar ju ALLTID en string
- dvs Typecasting är essientiellt för att kunna jobba med rätt datatyp

```
tal1 = input("Tal1:")
tal2 = input("Tal2:")
print(tal1+tal2)

tal1 = int(input("Tal1:"))
tal2 = int(input("Tal2:"))
print(tal1+tal2)
```

- andra typkonverteringar är
  a. float(<expression>)
  b. str(<expression>)

```
print('My age is: ' + 42)   # ERROR
print('My age is: ' + str(42))   # OK
```