Kickstart-kursus i programmering 2018

Martin Dybdal
dybber@di.ku.dk

DIKU
Københavns Universitet

20. august 2018

KØBENHAVNS
UNIVERSITET

De hele tal

- $\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots$
- også kendt som *integers*, *ints* eller *heltal*.
- I Python er der ingen grænser for hvor store tal der kan repræsenteres, ulig de fleste andre programmeringssprog.

KØBENHAVNS
UNIVERSITET

Operationer på heltal

Operation	**Syntaks**	**Præcedens**	**Associativitet**
Potensopløftning	`e1 ** e2`	1	Højre
Negation	`- e`	2	-

Multiplikation
Heltals division
Modulus
	`e1 * e2`
	`e1 / e2`
	`e1 % e2`
		3	Venstre

Addition Subtraktion	`e1 + e2` `e1 - e2`	4	Venstre

KØBENHAVNS
UNIVERSITET

Floating points (kommatal)

- $-3.0, -42.3, 0.0, 0.99999996, 13.27$
- også kendt som *floating points*, *floats*, *reals* (reelle tal).
- Aldrig helt præcise, så bør bruges med omtanke
- Python konverterer automatisk ints til floats, når de bruges i en kontekst hvor nødvendigt.

KØBENHAVNS
UNIVERSITET

Operationer på floats

Operation	**Syntaks**	**Præcedens**	**Associativitet**
Potensopløftning	`e1 ** e2`	1	Højre
Negation	`- e`	2	-
Multiplikation Division	`e1 * e2` `e1 / e2`	3	Venstre
Addition Subtraktion	`e1 + e2` `e1 - e2`	4	Venstre

KØBENHAVNS
UNIVERSITET

Boolske værdier

- `True`, `False`
- også kendt som *booleans*, *bools*, eller *sandhedsværdier*.

KØBENHAVNS
UNIVERSITET

Sammenlignings-operationer

Operation	Syntaks	Præcedens	Associativitet
Større end	`e1 >= e2`	5	Venstre
Mindre end	`e1 <= e2`	5	Venstre
Skarpt større end	`e1 > e2`	5	Venstre
Skarpt mindre end	`e1 < e2`	5	Venstre
Lighed	`e1 == e2`	5	Venstre
Ulighed	`e1 != e2`	5	Venstre

OBS! Sammenlign ikke lighed mellem floating points! Check om de er indenfor et interval.

```
>>> 0.1 == 1.0 - 0.9
False
```

KØBENHAVNS UNIVERSITET

Operationer på Boolske værdier

Operation	**Syntaks**	**Præcedens**	**Associativitet**
Logisk "negation"	`not e`	6	-
Logisk "og"	`e1 and e2`	7	-
Logisk "eller"	`e1 or e2`	8	-

KØBENHAVNS
UNIVERSITET

Strings

Værdier:

- `""`, `"hello world"`, `"linje 1\nlinje 2"`, `"\""`
- `''`, `'hello world'`, `'linje 1\nlinje 2'`, `'"'`

Operation	**Syntaks**	**Præcedens**	**Associativitet**
Konkatenering	`e1 + e2`	4	-

KØBENHAVNS
UNIVERSITET

Casting

Cast til int:

- `int(50) ==> 50`
- `int(7.6) ==> 7`
- `int("3") ==> 3`

Cast til float:

- `float(50) ==> 50.0`
- `float(7.3) ==> 7.3`
- `float("3") ==> 3.0`
- `float("4.2") ==> 4.2`

Cast til string:

- `str(50) ==> "50"`
- `str(7.3) ==> "7.3"`
- `str(True) ==> "True"`
- `str()` virker også på de fleste andre typer af værdier

KØBENHAVNS UNIVERSITET

Betingelser

Basalt brug:

```
if e:
    ... udføres hvis e evaluerer til True ...
else:
    ... udføres hvis e evaluerer til False
```

Udvidet:

```
if e1:
    ... udføres hvis e1 evaluerer til True ...
elif e2:
    ... udføres hvis e2 evaluerer til True
else:
    ... hvis hverken e1 eller e2 evaluerer til True ...
```

KØBENHAVNS
UNIVERSITET

Betingelser: Typiske skønhedsfejl

```
if x == True:
    ...
```

```
if x != True:
    ...
```

```
if not (x < 100):
    ...
```

KØBENHAVNS
UNIVERSITET

Variabler

- Variable kan bestå af bogstaver (a-z, A-Z), cifrene (0-9) eller underscore (_)
- Variable må ikke starte med et ciffer
- Må ikke navngives med et reserveret ord (fx `def`, `if`, `elif`, `else`, `global`, `True`, `False`, `return`, `and`, `or`, `not`, ...)
- Variabler er *case sensitive* (variablen `ABc` er forskellig fra `ABC`)
- Variabler defineres første gang de sættes

Eksempler:

```
foo = 17 + 42 / 6
bar = foo * 2.0
baz = bar < 10
```

KØBENHAVNS UNIVERSITET

Funktioner

- Funktioner følger samme navngivningsregler som variabler
- Du kan ikke have en variabel og en funktion der hedder det samme, da vil den senest definerede overskrive (de er i samme "navnerum")

Eksempel:

```
def kvadrat(x):
    return x * x

def kvadratsum(a, b):
    return kvadrat(a) + kvadrat(b)

kvadratsum(1+2, 4)
```

KØBENHAVNS
UNIVERSITET

Globale vs. lokale variabler

Eksempel 1:

```
foo = 42
def minfunktion():
    global foo
    foo = 20
minfunktion()
# variablen "foo" er nu sat til 20
```

Eksempel 2:

```
foo = 42
def minfunktion():
    foo = 20
minfunktion()
# variablen "foo" er stadig 42
```

Her er der to variabler med samme navn `foo`.

KØBENHAVNS
UNIVERSITET

Scoping, virkefelter

Hvilke variabler og funktioner er synlige hvor?

```
variabel1 = ...
```

A

```
def funktion1(argument0, argument1):
    variabel2 = ...
```

B

C

- `variabel1` er synlig i A, B, C
- `argument0` og `argument1` er synlige i B
- `variabel2` er synlig i B
- `funktion1` er synlig i B, C

KØBENHAVNS UNIVERSITET