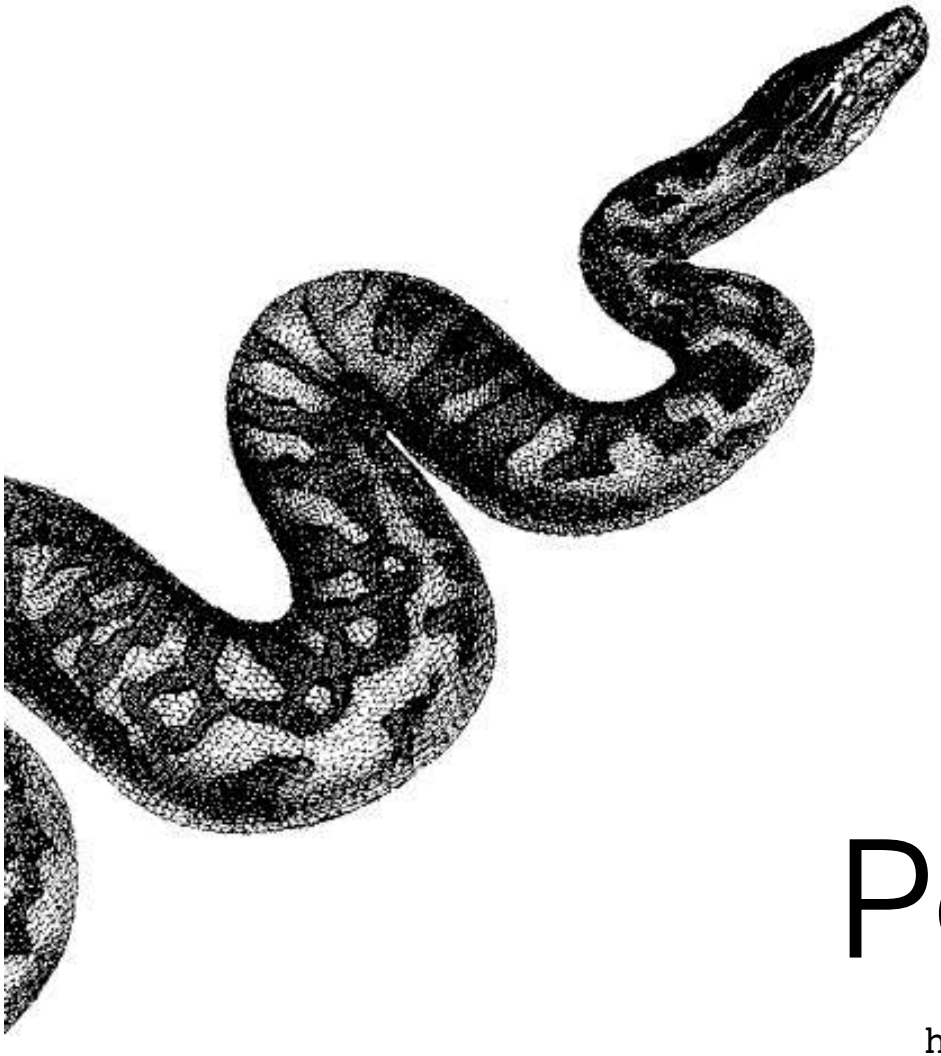
Python programmering

MacFest 2005



Per Tøfting

http://pertoefting.dk/macfest/

Indhold

Måder at afvikle et Python program på

Interaktiv:

Terminalen:

```
$ python
>>>
```

Afslut interaktiv og vend tilbage til alm. terminal vindue:

Ctrl+D

MacPython:

Menu: Windows -> Python Interactive

```
>>>
```

IDLE:

Starter med interaktiv vindue "Python Shell'

```
>>>
```

Måder at afvikle et Python program på

Modul filer:

```
$ python modul.py
```

Script filer:

Find evt. ud af hvad der skal stå i første linie af filen med:

```
$ which python
```

Første linie:

```
#!/usr/bin/python
```

eller:

```
#!/usr/bin/env python
```

Filen skal gøres eksekverbar:

```
$ chmod 744 script.py
```

eller:

```
$ chmod u+x script.py
```

```
$ ./script.py
```

Variabler

En variabel er en pegepind (reference), der peger på den adresse i computerens arbejdslager (RAM), hvor en værdi (objekt) er placeret.
Den eksiserer fra den tildeles en værdi. `a = 1`
Dens værdi kan ændres når programmet udføres. `a = 2`
Den kan tildeles værdier af forskellig type under udførelsen af programmet. `a = 'abc'`
Et objekts identitet: `id(a)`
Et objekts type: `type(a)`

Variabelnavne: starte med et bogstav (A til Z eller a til z) eller understregning (_) efterfulgt at nul eller flere bogstaver, tal (0 til 9) eller understregninger.

Data typer - tal

Heltal:

Decimal tal: `1, 23, 567, 99999999999L` (0 til 9)

Octale tal: `01, 027` (0 til 7, og starter med 0)

Hexadecimale tal: `0x1, 0xDA5` (0 til 9 og A til F, starter med 0x)

`sys.maxint` angiver maximale heltal. `L` eller `l` efter heltal angiver long version uden predefineret højeste tal.

Reelle tal (kommatal, floating point):

En sekvens af decimal tal der inkluderer et decimal punktum (.), en exponet del eller begge dele. `0.0, 1.0, 1.e2, 1e-2`

Komplekse tal:

Skrives som real-del `+` imaginære-del efterfulgt af `j` eller `J`.

`3+4J, 9+2J`

Aritmetiske operatorer

`+x`	Plus
`-x`	Minus
`x + y`	Addition
`x - y`	Subtraktion
`x * y`	Multiplikation
`x ** y`	Eksponent
`x / y`	Division, ægte
`x // y`	Division, heltals
`x % y`	Modulus, rest

Ved division brug: `from __future__ import division`

Data typer – sekvenser

Sekvens:

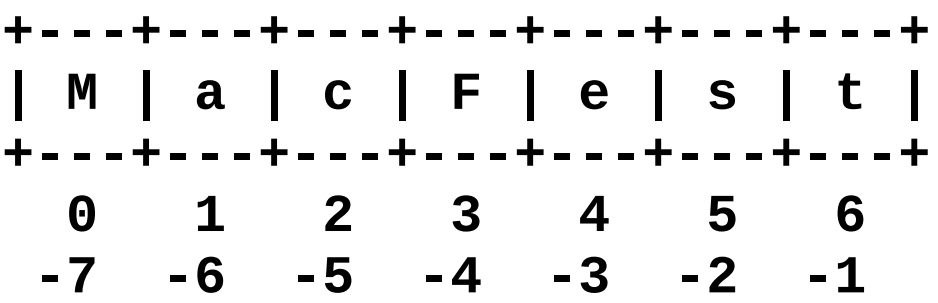
En ordnet kontainer af elementer, som er indexeret med ikke negative tal, starter fra 0.

Sekvenstyper:

Strenge (tekststrenge): `'Hej'`, `“MacFest”`

Tupler: `(2, 5)`, `(2, 'Hej', 0xA)`

Lister: `[2, 5]`, `[2, 'Hej', 0xA]`



Data typer – dictionary

Dictionary (ordbog):

En uordnet samling af elementer indexeret af næsten vilkårlige værdier kaldet nøgler.

Elementerne i et dictionary består af **nøgle:værdi** par.

Nøglerne kan være af forskellig type, men skal være hashable.

Værdierne er vilkårlige objekter og kan have forskellig type.

```
{'x':42,'y':3.14,'z':7}, {1:2,3:4}
```

Kontrol strukturer

Et programs flow er kontroleret af betingelser, løkker og funktionskald.

Betingelser (selektion):

- `if` sætninger
- `if else` sætninger
- `if elif ... else` sætninger

Løkker (iteration):

- `while` løkker
- `for` løkker

Funktionskald:

- `def` sætninger

if sætninger

Syntaks for `if` sætninger:

```
if udtryk:
   sætning(er)
```

```
if udtryk:
   sætning(er)
else:
   sætning(er)
```

```
if udtryk:
   sætning(er)
elif udtryk:
   sætning(er)
elif udtryk:
   sætning(er)
...
...
[else:
   sætning(er)]
```

Et udtryk er en sætning, der resulterer i en værdi, som evalueres til at være sand eller falsk i if sætninger og while løkker.

`pass` : den tomme sætning.

Relations- og betingelsesoperatorer

`x == y`	x er lig med y
`x != y`	x er ikke lig med y
`x < y`	x er mindre end y
`x > y`	x er større end y
`x <= y`	x er mindre end eller lig med y
`x >= y`	x er større end eller lig med y
`x is y`	x og y er det samme objekt
`x is not y`	x og y er ikke det samme objekt
`x in y`	x er et element i kontaineren y
`x not in y`	x er ikke et element i kontaineren y
`not x`	boolsk NOT
`x and y`	boolsk AND
`x or y`	boolsk OR

Returnerer `True` eller `False`.

Sammenligninger kan kædes sammen: `0 < a < 100`.

Sand eller falsk i boolks sammenhæng

I boolsk sammenhæng er enhver værdi enten sand eller falsk.

Falsk:

`False`
`None`

Tallet 0 uanset type

En tom streng, tuple, liste og dictionary

Sand:

`True`

Alle tal på nær 0

En ikke tom streng, tuple, liste eller dictionary

Brug: `if x:`

Ikke:
```
if x is True:
if x == True:
if bool(x):
```

while og for løkker

Syntaks for `while` løkke:

```
 while udtryk:
    sætning(er)
[else:
    sætning(er)]
```

`break`: bryd ud af løkke.

`continue`: sprig over resten af løkken, fortsæt med næste gennemløb af løkke.

Syntaks for `for` løkke:

```
 for element in itererbar:
    sætning(er)
[else:
    sætning(er)]
```

Funktioner

Syntaks for funktioner:

```
def funktionsnavn(parameterliste):
    sætning(er)

parameterliste:
    positions parametrer                w
    valgfrie parametrer                 x = udtryk
    ekstra positions parametrer         *y
    ekstra valgfrie parametrer          **z
```

Kald af funktion:

```
funktionsnavn(argumentliste)

argumentliste:
    positions argumenter
    navngivne argumenter
```

`return udtryk` : Retuner et resultat fra funktionen ellers returneres `None`.

Scope - Namespace

Scope:

Den del af et program hvor en variabels værdi kan tilgås.

Lokal variabel:

En variabel, som er oprettet i en funktions blok. Variablen er først tilgængelig, fra den er oprettet og tildelt en værdi.
Dens eksistens kendes ikke udenfor funktionen.
En funktions parametrer er også lokale variabler.
En lokal variabel med samme navn som en global variabel skjuler den globale variabel.

Global variabel:

En variabel der er oprettet udenfor funktionerne i et modul.
En global variabels værdi kan nås i en funktion, men skal værdien ændres skal variablen erklæres `global` i funktionen.

Moduler

```
import modulnavn [as nymodulnavn]
modulnavn.funktionsnavn()

from modulnavn import funktionnavn [as nyfunktionnavn]
from modulnavn import *
funktionsnavn()
```

Egne modulers pladsering

```
import sys
sys.path.append('/sti til modul(er)')
```

Eller brug af PYTHONPATH

`.bash_profile` tilføjes:

```
export PYTHONPATH=/sti til modul(er):$PYTHONPATH
```

Stien til et modul findes med Unix kommandoen `pwd`, når man står i biblioteket med modulet.

Exceptions - undtagelser

Syntaks for try/except sætninger:

```
 try:
    sætning(er)
 except [udtryk [, variabel]]:
    sætning(er)
[else:
    sætning(er)]
```

Syntaks for try/finally sætninger:

```
 try:
    sætning(er)
 finally:
    sætning(er)
```