INF100
moduler, standardbibliotek

David Grellscheid
Torstein Strømme

import og modules

```
# helpers.py

def spam(x):
    s = 'spam'
    return f'{s}, {s}, {s}, {x} and {s}.'

N_A = 6.02214e+23
```

```
# work1.py

import helpers

a = helpers.N_A
b = helpers.spam('eggs')
```

```
# work2.py

import helpers as h

a = h.N_A
b = h.spam('eggs')
```

```
# work3.py

from helpers import N_A, spam

a = N_A
b = spam('eggs')
```

```
# work4.py

from helpers import N_A as L, spam as foo

a = L
b = foo('eggs')
```

import og modules

```
# helpers.py

def spam(x):
    s = 'spam'
    return f'{s}, {s}, {s}, {x} and {s}.'

N_A = 6.02214e+23
```

```
import helpers
import math

a = helpers.N_A
b = helpers.spam('eggs')
c = 27
d = math.sin(math.pi)
e = a + c + d
```

(lokalt)	helpers	math
a	N_A	sin()
b	spam()	pi
c		cos()
d		sqrt()
e		…
		…
		…

<- ***namespace***

```
from helpers import N_A
from math import sin,pi

a = N_A
c = 27
d = sin(pi)
e = a + c + d
```

(lokalt)		
N_A		
sin	**ikke tilgjengelig**	
pi		
a		
c		
d		
e		

_ _name_ _

```
def show_name():
    return __name__   # *to* understrek hver side!

print('Navnet er’,show_name())
```

```
Navnet er __main__
```

_ _name_ _

```
# helpers.py

def show_name():
    return __name__    # *to* understrek hver side!
```

```
import helpers

print('Navnet er', helpers.show_name())
```

```
Navnet er helpers
```

```
import helpers as abc

print('Navnet er', abc.show_name())
```

```
Navnet er helpers
```

_ _name_ _

```
# something.py

def abc():
    return 77

def klm(x):
    return x+x

def xyz():
    return 'Hei'

if __name__ == "__main__":  # filen er brukt direkte
    # gjør noe
    print(abc())
    print(klm(12))
    # gjør noe annet
    # …
```

Om filen brukes som bibliotek med *import,* så kjører siste delen ikke.

_ _name_ _

```
def main():
    # …
    f3()
    # …
    f1()
    # …

def f1():
   return …

def f2():
    return …

def f3():
    return …

if __name__ == "__main__":
    main()
```

Kan skrive hovedprogram først i filen, før vi definerer andre funksjoner

dir() og *pydoc*

```
# helpers.py
"""
A module full of helpers. It’s really useful!
"""
def show_name():
    """Show the module’s name."""  # <- docstring
    return __name__   # *to* understrek hver side!
```

```
import helpers
print(dir(helpers))
```

```
[…,…,’show_name’]
```

liste av alle funksjoner, variabler, osv., som finnes i modulen

```
import helpers
help(helpers)
```

```
$ pydoc helpers
```

```
Help on module helpers:

NAME
    helpers - A module full of helpers. It’s really useful!

FUNCTIONS
    show_name()
        Show the module’s name.

FILE
    /Users/dg/INF100/exercises/l10/helpers.py

(END)
```

https://www.python.org/dev/peps/pep-0257/

tast 'q' for å quitte

Packages

Vi kan organisere moduler i mapper med bruk av en tom __init__.py fil:

```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/                  Subpackage for filters
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
```

```
import sound.effects as se

from sound.effects import echo

from sound.effects.echo import echo_maker
```

Standard Library

Stor utvalg:

* Regular expressions, `difflib`, `textwrap`
* `datetime`, `calendar`
* synchronized queue
* `copy`
* `math`, `decimal`, `fractions`, `random`
* `os.path`, `stat`, `tempfile`, `shutil`
* `pickle`, `sqlite3`, `zlib`, `bz2`, `tarfile`, `csv`
* Markup, internet protocols, multimedia, debugging, ...

https://docs.python.org/3/library

math

```
import math
```

- floor, ceil
- exp, log, log2, log10
- pow, sqrt
- sin, cos, tan
- pi, e

fractions

```
from fractions import Fraction
Fraction(16, -10)
Fraction(123)
Fraction()
Fraction('3/7')
Fraction(' -3/7 ')
Fraction('1.414213 \t\n')
Fraction('-.125')
Fraction('7e-6')
Fraction(2.25)
Fraction(1.1)
```

```
x = Fraction(3,4) + Fraction(1,6)
x
x.numerator
x.denominator
```

```
from math import pi
Fraction(pi)
Fraction(pi).limit_denominator(100)
Fraction(pi).limit_denominator(50)
```

datetime

```
from datetime import date
today = date.today()
my_birthday = date(today.year, 6, 24)

if my_birthday < today:
    my_birthday = my_birthday.replace(year=today.year + 1)

time_to_birthday = abs(my_birthday - today)
print(time_to_birthday)
```

```
from datetime import datetime
now = datetime.now()
exam = datetime(2020, 5, 28, 9, 0, 0)
print(exam - now)
```

calendar

```
import calendar
calendar.prmonth(2020,02)
```

itertools

```
import itertools
```

Iterator	Arguments	Results
`product()`	p, q, … [repeat=1]	cartesian product, equivalent to a nested for-loop
`permutations()`	p[, r]	r-length tuples, all possible orderings, no repeated elements
`combinations()`	p, r	r-length tuples, in sorted order, no repeated elements
`combinations_with_replacement()`	p, r	r-length tuples, in sorted order, with repeated elements
`product('ABCD', repeat=2)`		`AA AB AC AD BA BB BC BD CA CB CC CD DA DB DC DD`
`permutations('ABCD', 2)`		`AB AC AD BA BC BD CA CB CD DA DB DC`
`combinations('ABCD', 2)`		`AB AC AD BC BD CD`
`combinations_with_replacement('ABCD', 2)`		`AA AB AC AD BB BC BD CC CD DD`

random

```
import random
# random.seed(123456) # reproduserbare resultater

random.randint(1,6) #  1 <= N <= 6

random.choice('abcdef')
random.choice([11, 7.2, 'foo'])
random.choices('abcdefghi', k=100)

random.sample('abcdefghi', k=3)
```

```
random.random() # [0.0, 1.0)
random.uniform(12.0, 20.0) #  12.0 <= N <= 20.0

random.gauss(mu=40.0,sigma=12.0)
```

Eksterne pakker

flere 100.000 hos PyPI

http://pypi.python.org/pypi

...,Numpy, Scipy, Matplotlib, ...

Enkel installation med pip

Kvaliteten varierer!

Eksterne pakker

Installasjon med:

```
python -m pip install ...
```

"virtual environments" kan brukes til å separere ulike prosjekter som trenger eksterne pakker:

```
python -m venv create some_name  # 1. gang
source some_name/bin/activate    # etterpå
```