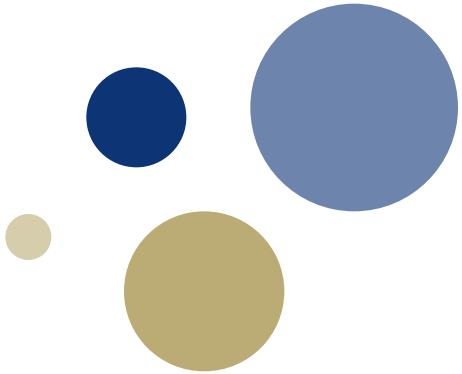
NTNU

Kunnskap for en bedre verden



Python: Funksjoner og moduler

Kapittel 5.7-5.10

TDT4110 IT Grunnkurs

Professor Guttorm Sindre

Læringsmål og pensum

- Mål
  - Kunne bruke bibliotek i Python, f.eks **random** og **math**
  - Kunne lage og kalle egne funksjoner med returverdi
  - Vite hvordan vi oppnår gjenbrukbare funksjoner
- Pensum
  - Starting out with Python: Chapter 5.7-5.10

  Value-Returning Functions and Modules

Bibliotek og import

Kapittel 5.7, 5.9

Standard Library-funksjoner

- Funksjoner i Pythons standardbibliotek
  - Kan brukes uten videre
- En ferdiglaget funksjon ≈ en svart boks
  - Vi kan bruke funksjonen uten å se inn i boksen
  - Vi har allerede brukt flere standardfunksjoner i Python, f.eks:
    - input( )
    - int( )
    - print( )
    - format( )
  - Vi har ikke behøvd å vite hvordan disse er laget
    - kan f.eks. bruke print() uten å skjønne dens 50+ kodelinjer

Funksjoner i andre bibliotek

- Funker også som ”svarte bokser”, for eksempel:
  - math.sqrt(x) # returnerer kvadratroten til x
  - cmath.sqrt(x) # kvadratroten til x (også for x < 0)
  - turtle.forward(p) # tegner en rett strek, p piksler lang
  - random.random() # gir en tilfeldig float i intervallet [0.0,1.0⟩
- Må importere biblioteksmodulen før bruk

```
import math
import random as r
x = 9
y = math.sqrt(x)
z = math.sin(x)
k = r.random()
```

```
from math import sqrt, sin
from random import random
x = 9
y = sqrt(x)
z = sin(x)
k = random()
```

```
from math import *
from random import *
x = 9
y = sqrt(x)
z = sin(x)
```

IKKE ANBEFALT!

For mer om fordeler og ulemper, se
http://stackoverflow.com/questions/710551/import-module-or-from-module-import

Modulen math - matematikk

- Innhold
  - Funksjoner for å utføre matematiske beregninger
  - Matematiske konstanter som pi og e
- Bruk av modulen:
  - Importere, f.eks. `import math`
  - Kalle funksjon, f.eks. `y = math.sqrt(x)`
  - Bruke konstant, f.eks. `omkr = 2 * math.pi * r`
- Oversikt over modulens innhold
  - Noen av funksjonene: Table 5-2 (p.237) i Gaddis-boka
  - Komplett liste: https://docs.python.org/3/library/math.html

Modulen random – tilfeldige tall

- Egentlig: pseudo-tilfeldige tall
- Diverse bruksområder, f.eks.
  - Spill og simuleringer
  - Statistiske analyser
  - Autogenerert musikk og kunst
- Noen funksjoner:
  - random.seed() # gi frøverdi (oftest ok uten)
  - random.random() # tilfeldig flyttall [0.0,1.0⟩
  - random.uniform(fra,til) # tilfeldig flyttall [fra, til⟩
  - random.randint(fra,til) # tilfeldig heltall [fra,til]
  - random.choice(liste) # ett tilfeldig element fra liste
  - random.randrange(fra,til,steg) # velger ett tall fra serien
    [fra, fra+steg, fra+2*steg, …, til⟩

NB! – [ ]

- Komplett liste av funksjoner:
  - https://docs.python.org/3/library/random.html

Void- og returverdi-funksjoner

- Noen funksjoner i Python returnerer ikke verdier, f.eks.
  - print() # skriver ut på skjermen
  - turtle.forward() # tegner en strek
  - random.seed() # gir frøverdi (”seed”) for random
  - Slike funksjoner kalles på engelsk *void*
- Andre funksjoner returnerer verdier, f.eks.
  - input() # returverdi: strengen som brukeren skrev
  - turtle.position() # returverdi (float, float): markørposisjon
  - random.randint(1,6) # returverdi int: en tilfeldig terningverdi 1-6
  - str.isdigit() # True hvis strengen str kun inneholder tall,
False hvis minst ett tegn er noe annet
- I Python oppgis ikke returtype ved definisjon av funksjon
  - I mange andre språk (f.eks Java, C, C++) oppgis dette

Skrive egne funksjoner med returverdi

Kapittel 5.8

Oppbygning av en funksjon

- Forelest forrige uke:
  - Funksjonsdefinisjonen begynner med ordet def
  - Parentes etter funksjonsnavnet har parametre
  - Kodeblokka til funksjonen har innrykk
    - Kan opprette lokale variable (for mellomregninger)
    - Kan bruke (+endre) globale variable (NB: sjelden anbefalt!)
- Nytt denne uka
  - Lage funksjoner som returnerer verdier
  - I så fall må return … inngå i kodeblokka
    - Typisk siste (nederste) setning i funksjonen
    - Kan stå andre / flere sted (f.eks. if-else-…)
  - Når return utføres, hopper kjøring tilbake der funksjonen ble kalt fra

```
def funksjonsnavn (para1, para2, para3):
    kode…
    kode…
    return uttrykk
```

Kan også returnere flere verdier

- Uttrykk bak return - hva som helst som gir en verdi
  - Kan returnere int, float, bool, str, liste osv.
- Kan også returnere flere verdier
  - Format: **return uttrykk1, uttrykk2**, … (komma mellom hver)
  - Antall variable i kallet må stemme, dvs.
    - En variabel på venstresiden av = for hver returnert verdi:

```
def get_name():
    first_name = input("First name? ")
    last_name = input("Last name? ")
    return  first_name,  last_name

# skriptet, der vi kaller funksjonen:
fornavn,  etternavn = get_name()
```

Hvorfor trenger vi returverdier?

- Funksjoner: Bruke samme kode flere sted i programmet
  - Eller i flere forskjellige programmer
- Funksjon må kommunisere med resten av programmet
- Tre mulige løsninger
  1. Ingen vesentlig kommunikasjon
     - Funksjonen leser selv sine inndata med input( )
     - Funksjonen skriver selv ut resultat med print( )
  2. Via globale variable
     - Før kallet puttes inndata til funksjonen i en global variabel
     - Funksjonen putter resultatet i en annen global variabel
     - Programmet henter resultatet fra sistnevnte variabel og bruker det
  3. Via parametre (inndata til funksjon) og returverdi (resultat)

Behovet for parametre og returverdier

- Funksjon må kommunisere med resten av programmet
- Tre mulige løsninger
  1. Ingen vesentlig kommunikasjon
     - Funksjonen leser selv sine inndata med input( )
     - Funksjonen skriver selv ut resultat med print( )
  2. Via globale variable
     - Før kallet puttes inndata til funksjonen i en global variabel
     - Funksjonen putter beregnet resultat i en annen global variabel, programmet bruker denne verdien videre
  3. Via parametre (inndata til funksjon) og returverdi (resultat)

DÅRLIG

Eksempel: dårlig vs. god

- Vi vil lage en funksjon som beregner absoluttverdi:
  - |x| = x , hvis x >= 0, |x| = −x , hvis x < 0
  - abs() fins allerede i standardbiblioteket
  - …men lager vår egen for illustrasjon
- Mulig bruk:
  - Statistiske analyser
  - Situasjoner der bare positive verdier gir mening

- Dårlig løsning 1: input() og print() i funksjonen
- Dårlig løsning 2: globale variable
- God løsning: parameter og return

kode: abso_v0.py abso_v1.py abso_v2.py

Oppgave: Lag deres egen funksjon

- Standardbiblioteket inneholder funksjonen round()
  - Ett argument: Avrunder til hele tall, round(2.87) → 3
  - Argument nr 2: ønsket antall desimaler
    - round(2.87, 1) → 2.9
    - round(12345.678, 2) → 12345.68
    - round(12345.678, –2) → 12300
  - En fin funksjon, men…
  - …følger ikke ”vanlige” avrundingsregler
    - round(2.5) → 2 # skulle vært 3
    - round(2.55) → 2.5 # skulle vært 2.6; round(2.75) → 2.8

- Vi vil lage en funksjon avrund() med ”vanlige” regler

Oppgave

avrL_v0.py

LETTERE:

Lag funksjon avrund() som tar ett argument (et flyttall) og som returnerer nærmeste hele tall. Halve skal konsekvent rundes oppover, f.eks.
2.5 → 3, 3.50 → 4

avrM_v0.py

MIDDELS:

Lag funksjon avrund() som tar to argumenter. Det første et flyttall som skal avrundes, det andre ønsket antall desimaler. Den andre parameteren skal ha defaultverdi 0 så funksjonen runder av til hele tall hvis det kun gis ett argument i kallet.

avrV_v0.py

VANSKELIGERE:

(a) I tillegg til MIDDELS, sørg for at avrund() også takler negative tall for det andre argumentet og kan avrunde grovere enn heltall, f.eks. avrund(12345.6, -2) skal gi 12300.

(b) Søk på nettet og finn ut hvorfor round() gjør "feilene"
(i) 2.5 → 2 og
(ii) 2.55 → 2.5
NB: (i) og (ii) har ulike grunner, prøv om du klarer å forstå begge!

HINT: int() gjør om flyttall til heltall, men bare kutter desimaldelen, runder aldri opp. Funksjonen må derfor gjøre litt i tillegg, enten en if-setning eller annet lite triks for at det rundes opp der dette er regelen.

Lagre funksjoner i moduler

Kapittel 5.10

Lagre funksjoner i moduler

- Modul : fil som inneholder Python-funksjoner
  - Programmer kan importere modulene og kalle funksjonene
    - import modulnavn
- Filnavn for modul skal slutte med .py
  - Ikke nøkkeluttrykk i Python (~~if.py, while.py,~~ …)
- Hvorfor moduler?: gruppere relaterte funksjoner
  - program blir lettere å forstå, teste og vedlikeholde
  - nyttige funksjoner kan brukes i mange program
  - EKSEMPEL: avrund() som vi nettopp lagde
    - generell bruksverdi, la oss putte den i en modul!
- NB! For at kodeeksemplene under skal virke, må fila avrunding.py ligge på samme katalog som ost.py, bank.py og skole.py når du kjører

kode: avrunding.py

bruk: ost.py, bank.py, skole.py

Oppsummering (1)

- Dette kapittelet dekket:
  - Returverdifunksjoner, inkludert
    - Å skrive returverdifunksjoner
    - Å bruke returverdifunksjoner
    - Å returnere flere verdier fra en funksjon
  - Å bruke bibliotekfunksjoner og import-uttrykket
  - Moduler, inkludert
    - Modulene random og math
    - Å gruppere dine egne funksjoner i moduler

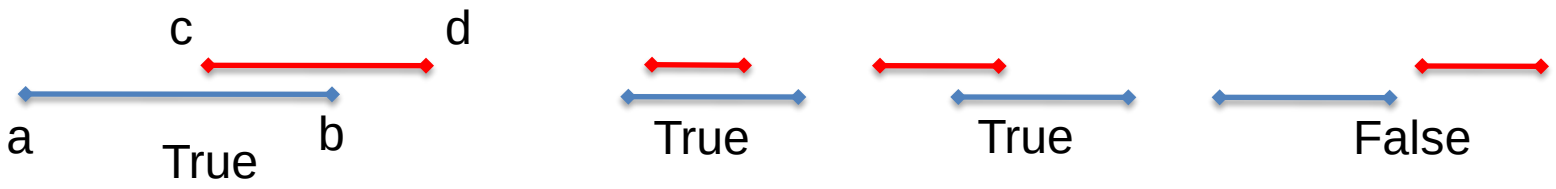
Oppsummering (2), mange slags funksjoner

	Innebygde	I bibliotek	Egendefinerte
Uten returverdi - eksempel:…	print( )	color()	abso_v0()
- hvordan kalle? Tilsvarer hel setning	print('Hei!')	turtle.color('red')	abso_v0()
Med returverdi - eksempel: …	abs() input()	random() sin()	abso_v2()
- hvordan kalle? Tilsvarer en verdi	a=input('Tall? ') print(abs(a))	x=random.random() y=math.sin(x)/2	x=abso_v2(-5.1)
Hva må skrives lenger oppe i programmet?	ingenting	import turtle import random import math eller… from … import … (slippe prefiks ved kall)	def funknavn(par): setninger …. #hvis returverdi **return** verdi

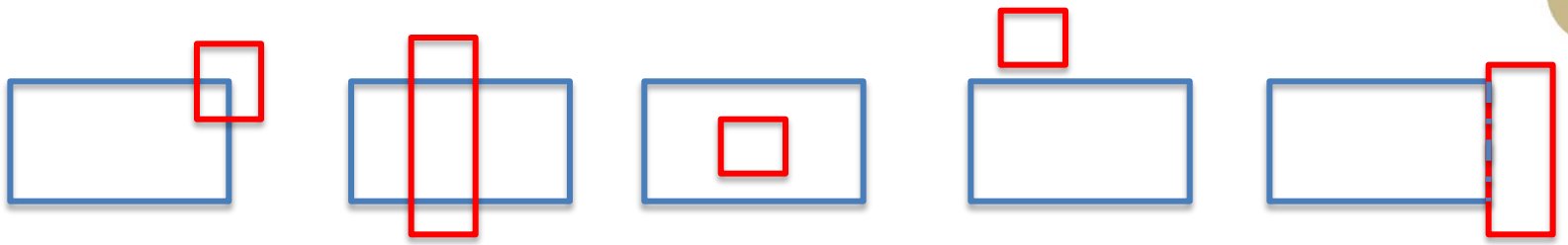
Ekstraoppgave, del 1

- Lag funksjonen overlapp_intervall(a,b,c,d)
  - Gitt intervallene (a,b) og (c,d): overlapper de hverandre?
  - Returnere True hvis overlapp, ellers False
  - Om ett begynner der det andre slutter: ikke overlapp
- Bruk parametre og returverdi
  - IKKE global variabel, IKKE input og print i funksjonen

kode: overlapp_intervall_v1.py …v2 …v3



Ekstraoppgave, del 2



- Gitt to rektangler parallelle med x- og y-aksen: Overlapper de?
- Lag en funksjon som sjekker dette!
  - 8 parametre, x1,y1, x2,y2, x3,y3, x4,y4
    - (x1,y1) og (x2,y2) er koordinatene for nedre venstre og øvre høyre hjørne i ett rektangel
    - (x3,y3) og (x4,y4) er tilsvarende koordinater for det andre rektanglet
- Dette er ganske tricky!
  - Ligner på oppgave 2d, eksamen i TDT4102 (C++), juni 2015
  - Potensielt jobbintervjuspørsmål…
- Blir vesentlig enklere med gjenbruk av **overlapp_intervall( )**
  - Bare vi skjønner hvordan vi skal bruke den…

kode: overlapp_rektangel_halvferdig.py
…_ferdig.py ; …_import.py