NTNU
Det skapende universitet

TDT4110 Informasjonsteknologi grunnkurs:
Tema: Funksjoner med retur og moduler

- 3rd edition: Kapittel 5.7-5.10

Professor Alf Inge Wang

Læringsmål og pensum

- Mål
  - Beherske returverdier og returverdifunksjoner
  - Bruke matematikkbibliotek i Python
  - Generere tilfeldige tallverdier
- Pensum
  - "Value-Returning Functions and Modules"
  - Starting out with Python 3rd edition: Chapter 5.7-5.10

NTNU
Det skapende universitet

www.ntnu.no

Intro til returverdifunksjoner: Generering av tilfeldige tall

Kapittel 5.7

NTNU
Det skapende universitet

www.ntnu.no

Funksjoner

- Enkel funksjon:
  grupperte uttrykk for å utføre en viss oppgave
- Returverdifunksjoner:
  Tilsvarende ovenfor, men returnerer en verdi

Standard Library-funksjoner og import-uttrykket

- Standardbibliotek: bibliotek inneholdende forhåndskrevne funksjoner som kommer med Python
  - Bibliotekfunksjoner utfører oppgaver som programmerere ofte trenger
  - Eks: print, input, range
  - Betraktes gjerne (men ikke nødvendigvis(!)) som «black box»
- Noen bibliotekfunksjoner er bygd inn i Python-tolkeren
  - Bare kall funksjonen for å bruke den

Standard Library-funksjoner og import-uttrykket (forts.)

- En funksjon inne i et bibliotek, kan sees på som en svart boks, dvs. at vi bruker funksjonene uten å vite hva som foregår inne i den svarte boksen.

Å generere tilfeldige tall

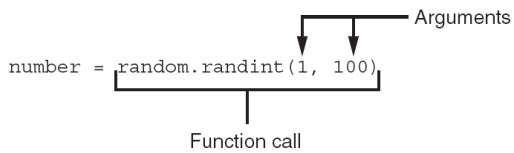
- Tilfeldig genererte tall er nyttige i mange programmeringsøyemed
- Random-modulen: inneholder bibliotekfunksjoner til arbeid med tilfeldige tall
- «Dot notation»: notasjon for å kalle metoder som tilhører en gitt modul:
  - Format: module_name.function_name()
  - Eks: random.random()

Å generere tilfeldige tall (forts.)

- Funksjonen `randint()`:
  - Genererer et tilfeldig heltall i intervallet spesifisert gjennom *argumentet*
  - Returnerer det tilfeldige tallet til en annen del av programmet som *kalte* på denne funksjonen
  - Det returnerte tallet kan brukes hvor som helst et heltall kan brukes
  - Dette kan du eksperimentere med i *interaktiv modus*

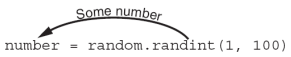
Å generere tilfeldige tall (forts.)

- Kall av random-funksjon med to argumenter:
  - To input argumenter
  - Returnerer tilfeldig heltall som typisk lagres i en variabel



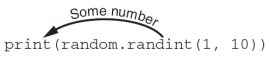
Å generere tilfeldige tall (forts.)

Retur av verdi fra random-funksjon:



A random number in the range of 1 through 100 will be assigned to the `number` variable.

Vise et tilfeldig tall i konsollet vha. random-funksjon:



A random number in the range of 1 through 10 will be displayed.

Å generere tilfeldige tall

- Funksjonen `randrange`: tilsvarende funksjonen range, men returnerer et tilfeldig valgt heltall fra den resulterende sekvensen:
  - Samme argumenter som for funksjonen range
- Funksjonen `random`: returnerer et tilfeldig flyttall i intervallet [0.0,1.0)
- Funksjonen `uniform`: returnerer et tilfeldig flyttall, men tillater brukeren å spesifisere et intervall

Seeds for tilfeldige tall

- Tilfeldige tall laget av funksjoner i modulen random er egentlig *pseudo-tilfeldige* tall
- *Seed*-verdi: initialiserer formelen som genererer de tilfeldige tallene
  - Du trenger å bruke forskjellige *seeds* for å få forskjellige serier med tilfeldige tall
    - *Default*-verdien for seeds er systemtiden
    - Du kan bruke funksjonen `random.seed()` for å sette ønsket *seed*-verdi

Skrive egne funksjoner med retur

Kapittel 5.8

Å skrive egne returverdi-funksjoner

- For å skrive en returverdifunksjon skriver du en *enkel funksjon* og legger til en eller flere return-uttrykk
  - Format: `return` uttrykk
    - Verdien til uttrykk vil returneres til den delen av programmet som kalte på funksjonen
    - Uttrykket i utrykket `return` kan være et komplisert uttrykk som summen av to *variable* eller resultatet av en annen returverdifunksjon

Oppbygning av en funksjon.

Når man definerer/lager en funksjon bruker man det reserverte ordet `def`

All koden i funksjonen (kroppen) skrives i *innrykk*.

Avslutter med `return` (når noe skal returneres)

Funksjoner kan også skrives i den interaktive omgivelsen

Navn på funksjon

Parametere

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

Funksjonskropp

Avslutning

17

Kall av funksjoner

Ved kall av en funksjon brukes *funksjonsnavnet*, samt en liste av *argumenter* (verdier) som matcher navnet på funksjonen og lista av parametere:

Parametere

```
def arealRektangel (b,h):
    # Beregne areal av rektangel
    return b*h

>>> arealRektangel(20,4)
80
```

Argumenter

Resultatet

19

Å bruke IPO-kart

- IPO-kart: beskriver input (inndata), processing (prosessering) og output (utdata) til en funksjon
  - Verktøy for å designe og dokumentere funksjoner
  - Typisk presentert i kolonner
  - Har normalt korte beskrivelser av inndata, prosessering og utdata uten å gå i detalj
    - Inkluderer ofte nok informasjon til å erstatte et flytdiagram

20

Å bruke IPO-kart (forts.)

IPO Chart for the `get_regular_price` Function		
Input	Processing	Output
None	Prompts the user to enter an item's regular price	The item's regular price

```
def get_regular_price():
    price=int(input("Regular price? "))
    return price
```

Å bruke IPO-kart (forts.)

IPO Chart for the `discount` Function		
Input	Processing	Output
An item's regular price	Calculates an item's discount by multiplying the regular price by the global constant `DISCOUNT_PERCENTAGE`	The item's discount

```
DISCOUNT_PERCENTAGE=0.20

def discount(price):
   discount = price*DISCOUNT_PERCENTAGE
   return discount
```


Å returnere *boolske verdier*

- Boosk funksjon: returnerer enten `True` eller `False`
  - Brukes til å teste et kriterium for beslutningsstrukterer og repetitive strukturerer
    - F.eks å repetere handlingen i en løkke til en teller har nådd et visst tall
    - Vanlige beregninger, som å sjekke om et tall er partall kan enkelt repeteres gjennom å kalle på en funksjon
  - Brukes til å forenkle sjekk av inndata for gyldighet
    - Du kan spørre om verdier, og bare godta enkelte verdier ved å returnere `True` hvis inndata er en av disse
- Vi ser på et eksempel!

kode: tre_gangen.py

Å returnere flere verdier

- I Python kan en funksjon returnere flere verdier
  - Spesifisert etter return-uttrykket, atskilt av kommaer
    - Format: `return uttrykk1, uttrykk2`, etc.
  - Når du kaller på en slik funksjon i et tilordningsuttrykk trenger du en egen variabel på venstresiden av = operatoren for å ta i mot hver returnert verdi:

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

first_name, last_name = get_name()
```

kode: names.py

Modulen math

- Modulen math: er en del av *standard library* som inneholder nyttige funksjoner for å utføre matematiske beregninger
  - Typisk akseptere en eller flere verdier som argumenter, deretter å utføre matematiske operasjoner og returnere resultatet
  - Bruk av modulen betinger et import math-uttrykk

Funksjoner i Modulen math:

`math` Module Function	Description
`acos(x)`	Returns the arc cosine of `x`, in radians.
`asin(x)`	Returns the arc sine of `x`, in radians.
`atan(x)`	Returns the arc tangent of `x`, in radians.
`ceil(x)`	Returns the smallest integer that is greater than or equal to `x`.
`cos(x)`	Returns the cosine of `x` in radians.
`degrees(x)`	Assuming `x` is an angle in radians, the function returns the angle converted to degrees.
`exp(x)`	Returns e^x
`floor(x)`	Returns the largest integer that is less than or equal to `x`.
`hypot(x, y)`	Returns the length of a hypotenuse that extends from (0, 0) to (`x`, `y`).
`log(x)`	Returns the natural logarithm of `x`.
`log10(x)`	Returns the base-10 logarithm of `x`.
`radians(x)`	Assuming `x` is an angle in degrees, the function returns the angle converted to radians.
`sin(x)`	Returns the sine of `x` in radians.
`sqrt(x)`	Returns the square root of `x`.
`tan(x)`	Returns the tangent of `x` in radians.

Modulen math (forts.)

- Modulen math definerer variablene pi og e, som er tilordnet de matematiske verdiene for *pi* og *e*
  - Kan brukes i likninger som fordrer disse verdiene for å få mer nøyaktige resultater
- Variabler må også kalles på ved å bruke *dot*-notasjonen:
  - Eks:
    - `circle_area = math.pi * radius**2`

Lagre funksjoner i moduler

Kapittel 5.10

Å lagre funksjoner i moduler

- I større, kompliserte programmer er det viktig å organisere koden godt
- Modularisering: å gruppere relaterte funksjoner i moduler
  - Gjør at programmene er lettere å forstå, teste og vedlikeholde
  - Gjør det lettere å bruke gode om igjen for flere programmer
    - Importer modulen som inneholder den fordrede funksjonen til hvert program som trenger den

Å lagre funksjoner i moduler (forts.)

- En Module er en fil som inneholder Python-kode
  - Inneholder funksjonsdefinisjoner, men inneholder ikke kall til disse funksjonene
    - Programmer vil importere modulene og kalle funksjonene
- Regler for modulnavn:
  - Filnavn skal slutte med .py
  - Kan ikke være det samme som et nøkkeluttrykk i Python
- Importer moduler med import-uttrykket

Funksjoner som bruker funksjoner – et større eksempel

Vi skal lage funksjonen `romertall` som tar inn et tall mellom 0 og 9 og returnerer et romertall som svarer til tallet.

Romertallene fra 0 til 9 er som følger:

```
0='', 1=I, 2=II, 3=III, 4=IV, 5=V, 6=VI, 7=VII,
  8=VIII, 9=IX
```

Lag også koden slik at en bruker kan skrive inn et tall og få oversatt det til et romertall samt skrive det ut til konsollet så lenge at tallet brukeren skriver inn er større enn 0.

Benytt `def`, `if`, `elif`,`else`,`input`,`print`,`while` og `return`.

romertall_1.py

Funksjoner som bruker funksjoner – et større eksempel (fortsettelse)

For å angi det siste sifferet i et heltall som et romertall bruker vi symbolene I=1 og V=5

For å angi det nest siste sifferet i et heltall som et romertall bruker vi symbolene X=10 og L = 50

For å angi det nest-nest siste sifferet i et heltall som et romer tall bruker vi symbolene, C=100 og D = 500 etc

Romertall oppfører seg på samme måte for tall større en 10, men da med brukes X, L og C brukes i stedet for I, V og X

Funksjoner som bruker funksjoner – et større eksempel (fortsettelse)

Vi skal utvide slik at vi kan skrive et romertall avhengig av hvilket siffer vi skal angi.

Lag funksjonen `romersiffer` som tar inn 4 parametere:

- n: heltall mellom 0 og 9 som skal oversettes til romertall.
- onechar: Angir hvilket romertall som skal brukes for enerplassen
- fivechar: Angir hvilket romertall som skal brukes for femmerplassen
- tenchar: Angir hvilket romertall som skal brukes for tierplassen

Bruk:

For å angi bakerste siffer vil du gjøre følgende funksjonskall:

```
r = romersiffer(n,'I','V','X')
```

For å angi neste bakerste siffer vil du gjøre følgende funksjonskall:

```
r = romersiffer(n,'X','L','C')
```

Benytt `def`, `if`, `elif`, `print` og `return`.

romertall_2.py

Funksjoner som bruker funksjoner – et større eksempel (fortsettelse)

Endre funksjonen `romertall` slik at man kan skrive inn et tall mellom 0 og 999 og får returnert et riktig romertall.

Den nye funksjonen må plukke ut siffer for siffer og bruke funksjonen `romersiffer` for å generere riktig romertall:

Sjekke om tallet har siffer på hundreplassen -> C/D/M

Sjekke om tallet har siffer på tierplassen -> X, L, C

Sjekke om tallet har siffer på enerplassen -> I, V, X

Bruk:

```
r = romertall(n)
```

Benytt `def`, `if`, `elif`, `print` og `return`, samt heltallsdivisjon og restdivisjon for å gå igjennom ulike siffer.

romertall_3.py

NTNU
Det skapende universitet

Oppsummering

- 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

NTNU
Det skapende universitet