Programmering og Modellering (PoM)
Ugeseddel 2 — Uge 37 — Deadline 12/9

Kim Steenstrup Pedersen, Katrine Hommelhoff Jensen, Knud Henriksen,
Mossa Merhi og Hans Jacob T. Stephensen

4. september 2014

1 Plan for ugen

På dette kursus beskrives fire måder, hvorpå man i `python` kan bryde den normale sekventielle udførelse af programsætninger:

funktionskald Kald af en funktion (og af en metode) bevirker udførelse af funktionens (metodens) krop.

valg og forgreninger I forbindelse med betingelser kan man med `if`, `else` og `elif` styre til- og fravalg af programdele.

iteration Programdele kan gentages, styret af en betingelse (`while`) eller en liste (`for ... in`).

undtagelser På tværs af de tre ovennævnte "strukturerede" programkonstruktioner kan man kaste (`raise`) og gribe (`try ... except`) undtagelser.

Den centrale mulighed for at samle en programdel under et enkelt navn ved at definere (`def`) den som en *funktion* gennemgåes i denne uge.

I denne uge fortsættes også med gennemgangen af de forskellige muligheder for valg og forgreninger, og gentagelseskonstruktionerne `while` (med de tilhørende afbrydelser `break` og `continue`) og `for`. Senere i kurset vil vi gennemgå undtagelser og hvorledes disse kan gribes og håndteres.

Desuden omtales datatypen strenge og indlæsning til programmer fra tastatur, samt introduktion til forskellige talsystemer og boolsk algebra.

Forberedelse til forelæsningerne:

Til tirsdag:
Læs Guttag kap. 2.3 - 2.4, 3 - 3.2, 4 - 4.2. Gennemlæs også noterne om positionsrepærsentation af Nils Andersen og Talrepræsentation af Klaus Grue.

Til torsdag:
Læs Guttag kap. 2.3 - 2.4, 3 - 3.2, 4 - 4.2, samt noterne om om positionsrepærsentation af Nils Andersen og Talrepræsentation af Klaus Grue.

Noterne findes via punktet "Undervisningsmateriale" på kursushjemmesiden.

1.1 Individuel opgave

Den 12/9 senest klokken 15:00 skal besvarelse af følgende opgave afleveres elektronisk via Absalon. Opgaven skal besvares individuelt og skal godkendes, for at du kan kvalificere dig til den afsluttende tag-hjem eksamen. Opgavebesvarelsen skal oploades via kursushjemmesiden på Absalon (find underpunktet `ugeseddel2` under punktet `Ugesedler og opgaver`). Kildekodefiler ("script"-filer) skal afleveres som "ren tekst", sådan som den dannes af `emacs`, `gedit`, `Notepad`, `TextEdit` eller hvilket redigeringsprogram man nu bruger (*ikke* PDF eller HTML eller RTF eller MS Word-format). Filen skal navngives *fornavn.efternavn*.37.py, mens andre filer skal afleveres som en PDF-fil med navnet *fornavn.efternavn*.37.pdf.

2i1 Denne opgave omhandler rødder i komplekse andengradspolynomier.

Theorem 1 (TL 3.4.5) *Lad $f : \mathbb{C} \to \mathbb{C}$ være givet ved $f(z) = az^2 + bz + c$ med $a, b, c \in \mathbb{C}$ og $a \neq 0$. Da har andengradsligningen $f(z) = 0$ løsningerne*

$$z = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \tag{1}$$

hvor $\sqrt{b^2 - 4ac}$ er den komplekse kvadratrode med argument i $[0, \pi)$.

Theorem 2 (TL 3.4.2) *Lad $0 \neq z = re^{i\theta} \in \mathbb{C}$. Da har z to kvadratrødder w_1, w_2 givet ved*

$$w_1 = \sqrt{r}e^{i\theta/2}, \quad w_2 = \sqrt{r}e^{i(\theta+2\pi)/2}$$

Python har en indbygget datatype til repræsentation af komplekse tal, kaldet `complex`. Dertil findes der et bibliotek med mathematiske funktioner på komplekse tal kaldet `cmath`. Vi skal dog prøve at skrive nogle af funktionerne selv.

a) Skriv funktionerne `cmod(z)` og `carg(z)`, der tager det komplekse tal z som parametre og returnerer hhv. modulus og argumentet af det komplekse tal z. Det er ikke tilladt at benytte `cmath.polar` eller `cmath.phase`.

b) Skriv funktionerne `csqrt1(z)` og `csqrt2(z)` der tager det komplekse tal z som parameter og beregner kvadratrøderne vha. thm. 2. Brug funktionerne fra delopgave (a). [Note: Ovenstående funktioner kan skrives mere elegant ved hjælp af Pythons sæt (eng. tuple) datastruktur.]

c) Brug thm. 1 til at skrive funktionerne `polyrod1` og `polyrod2`, der bestemmer rødderne givet et komplekst polynomium $f(z)$. Du bestemmer selv hvilke parametre de to funktioner skal tage. Det er tilladt at bruge biblioteket cmath, men funktionerne fra (a) og (b) bruges i stedet for `cmath.polar, cmath.sqrt`.

d) Alle funktioner ovenfor skal dokumenteres med docstrings i forbindelse med funktionsdefinitionen.

1.2 Tirsdagsøvelser

Besvarelser af disse opgaver skal ikke afleveres, men opgaverne forventes løst inden torsdag.

2ti1 Mange beregninger forløber på den måde, at man i langt de fleste tilfælde kan følge en *hovedregel*, som der i nogle få tilfælde er *undtagelser* fra. For at programmere sådan en beregning kan man begynde med at teste for undtagelserne og derefter, hvis ingen af disse foreligger, bruge hovedreglen.

Et eksempel med denne struktur er flertalsdannelse af engelske navneord. Som hovedregel danner navneord flertal på engelsk ved tilføjelse af *-s*. Som en forsimplende antagelse vil vi i denne opgave gå ud fra, de eneste undtagelser er *man*, *woman*, *sheep* og *mouse*, som i flertal henholdsvis hedder *men*, *women*, *sheep* og *mice*.

Skriv et Python program (script), der virker på følgende måde, når det køres: Programmøren angiver ordet vi ønsker at kende flertal på ved at angive det i en variabel. Derpå udskrives teksten `Det engelske navneord` , efterfulgt af det ønskede ord. Programmet skriver derpå `hedder` *flertalsform* `i flertal.`, hvor *flertalsform* er flertalsformen af det angivet ord. Programmet terminere derpå.

Her er et par eksempler på kørsel af programmet:

```
Det engelske navneord ship
hedder ships i flertal.
Det engelske navneord sheep
hedder sheep i flertal.
Det engelske navneord lady
hedder ladys i flertal.
```

2ti2 Skriv et Python program (script) som undersøger tre variable x, y og z for derpå at printe det største ulige tal blandt dem. Hvis ingen af dem er ulige, skal programmet printe en besked ud om dette.

2ti3 Et tal a er en potens af b, hvis det kan divideres med b (dvs. b går op i a) og a/b er en potens af b. Dvs. vi kan skrive følgende

$$a = b^n \Rightarrow a/b = b^{n-1} \ .$$

Skriv et program som definere to variable a og b og udskriver en besked hvis a er en potens af b. Hvis ikke, skal programmet udskrive en besked om at det ikke er tilfældet. Prøv forskellige værdier for a og b (både hvor a er en potens af b hvor det ikke er tilfældet) og afprøv om dit program fungere som det skal for disse værdier.

2ti4 (Hint: Denne opgave er god at løse inden du løser den obligatoriske opgave):

Python har en indbygget datatype til repræsentation af komplekse tal, kaldet `complex`. Vi kan oprette det komplekse tal $z = 3 + i4$ således `z = complex(3, 4)`. Prøv at skrive `help(complex)` i Python fortolkeren og gør dig bekendt med tilladte operationer for `complex`. Prøv at lave følgende beregninger i hånden og derpå tjek dem med Python:

$$\begin{aligned} z &= 3 + i4 \\ c &= 7 - i2 \\ z^2 &= ? \\ cz &= ? \\ 3 * c &= ? \\ c + z &= ? \end{aligned}$$

1.3 Torsdagsøvelser

Besvarelser af disse opgaver skal ikke afleveres, men opgaverne forventes løst inden tirsdag i efterfølgende uge.

2to1 Fortsæt tabel 1 i noterne om positionsrepræsentation fem pladser længere. Udtryk decimaltallet 100 i binær notation. Hvad er værdien af `0145` i Python? Hvad er værdien af `0xe6`? Hvad er decimalværdien af det binære tal 10011?

2to2 Antag at følgende variable har følgende boolske værdier:

```
A = True
B = False
C = True
```

Prøv at beregne værdierne af følgende udtryk i hånden:

```
not (A and B)
(A and C) or B
A and B or A
A and not C
```

Tjek dine resultater ved hjælp af Python fortolkeren.

2to3 Lad os fortsætte med tirsdagsopgave 2ti1 ved at omskrive den samt at tillade input fra brugeren af programmet.

Definer en funktion `flertal()` uden formelle parametre, der virker på følgende måde, når den kaldes: Først udskrives teksten `Det engelske navneord` , og derefter venter funktionen på inddata. Hvis brugeren indtaster ordet `exit`, afsluttes funktionen, men ellers skriver den `hedder` *flertalsform* `i flertal.`, hvor *flertalsform* er flertalsformen af det indtastede ord, og fortsætter med endnu en gang at spørge `Det engelske navneord` .

Her er et eksempel på funktionskald:

```
flertal()
Det engelske navneord ship
hedder ships i flertal.
Det engelske navneord sheep
hedder sheep i flertal.
Det engelske navneord lady
hedder ladys i flertal.
Det engelske navneord exit
```

2to4 Med et heltal $g \geq 2$ som *grundtal* fremstilles heltallet n som cifferfølgen

$$b_{k-1}\ b_{k-2}\ \ldots\ b_2\ b_1\ b_0$$

med "cifre" b_i, $0 \leq b_i \leq g-1$ for $0 \leq i \leq k-1$, givet ved

$$\begin{aligned} b_0 &= n \texttt{ \% } g \\ b_1 &= n \texttt{ // } g \texttt{ \% } g \\ b_2 &= n \texttt{ // } g \texttt{ // } g \texttt{ \% } g \\ &\vdots \end{aligned}$$

Definer en funktion `reprBase(n,g)` af to formelle parametre, sådan at et kald af form `reprBase(`n`,`g`)` på en linje udskriver n som cifferfølge med grundtal g, idet de enkelte "cifre" adskilles af blanktegn. [Vink: Hvis $n < g$, skal funktionen blot udskrive n, og ellers skal cifferfølgen for n `//` g efterfølges af cifferet n `%` g.]

Eksempler:

```
reprBase(61501531,100)
61 50 15 31
reprBase(416,3)
1 2 0 1 0 2
```

Sørg for, cifrene kommer i den rigtige rækkefølge (de mest betydende forrest; mindst betydende ciffer til sidst).

Hvad gør din funktion, hvis den kaldes med g `= 1`? Hvorfor?

2to5 Lav en implementation af kvadratrodsalgoritmen vi gennemgik i uge 1 — algoritmen er beskrevet i bogen samt på slides. Lav din implementation som en funktion. Afprøv derpå din kvadratrodsalgoritme og sammenlign den med `math.sqrt` ved at udskrive en tabel som denne:

```
1.0 1.0          1.0          0.0
2.0 1.4142   1.4142   2.220e-16
3.0 1.7320   1.7320   0.0
4.0 2.0            2.0          0.0
...
```

Hvor første søgle angiver tallet vi tager kvadratrod af, anden søgle angiver kvadratrodresultatet produceret af din funktion, tredje sølge angiver kvadratroden produceret med `math.sqrt`, og sidste søgle angiver differencen mellem de to metoders kvadratrødder. Fortsæt tabellen op til eksempelvis 9.0.

2to6 Den indbyggede funktion `eval` tager som parameter en streng og evaluerer strengen med Python fortolkeren og returnere resultatet.

```
>>>eval('1+2*3')
7
>>>import math
>>>eval('type(math.pi)')
<type 'float'>
```

Skriv en funktion kaldet `eval_loop` som iterativt beder brugeren indtaste en streng som derpå evalueres med `eval` og udprinter resultatet. Funktionen skal fortsætte indtil brugeren skriver `'done'`.

2to7 Definer en funktion `tobogst(`w`)` med følgende virkning: Hvis w er et ord med n bogstaver, giver funktionskaldet `tobogst(`w`)` anledning til udskrivning af de $\binom{n}{2}$ to-bogstavs-ord, som kan dannes af w (med de to bogstaver i samme rækkefølge som den, hvori de forekom i w). Eksempel: Funktionskaldet `tobogst('arme')` kan give udskriften

```
ar am ae rm re me
```

2to8 Antallet af delmængder med netop k elementer udtaget fra en mængde med n forskellige elementer er $\binom{n}{k}$ (læses: "binomialkoefficienten n over k" eller blot "n over k") og kan beregnes af

$$\binom{n}{k} = \begin{cases} 0 & \text{hvis } k < 0 \text{ eller } k > n \\ 1 & \text{hvis } k = 0 \text{ eller } k = n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{for } 0 < k < n \end{cases} \tag{2}$$

Definer en funktion `binomcoeff(n,k)`, som ud fra sammenhængen i (2) beregner og returnerer binomialkoefficienten n over k.