

Studieretningscase

Matematik & Programmering

Indholdsfortegnelse:

- Introduktion
- Problemstilling
- Teori
 - **Matematik**
 - Hvad er renters renter og rekursionsligninger
 - Beregning af renters renter med rekursionsligninger
 - Talfølger
 - Eksponentielle funktioner
 - Programmering i matematik
 - **Programmering**
 - Variabler
 - Funktioner
 - Input og Buttons
 - Betinget udførsel
 - Matematik i Programmering
- Vurdering
- Konklusion
- Litteraturliste & kilder
- Bilag

Introduktion:

Renter kan både være sjovt og knapt så sjovt. Det hele kommer an på om du har penge i banken eller om du skylder penge. For ved positiv renters rente vil beløbet stige eksponentielt. Det kan være svært at forestille sig den eksponentielle stigning heldigvis har vi matematik og programmering til at hjælpe med dette. Når vi unge bliver ældre kan det også være super vigtigt at kende til renters, da vores liv kommer til at være præget af det.

Problemstilling:

- **Hvad er renters renter og hvordan regnes det i fagene?**

Underspørgsmål/underemner:

- Hvad er rekursionsligninger?
- Hvordan hænger rekursionsligninger og renters renter sammen?
- Hvordan kan man bruge programmering som et værktøj til at løse renters renter?
- Hvordan hænger programmering og matematik sammen?

Uddybelse:

Jeg vil med denne case forklare og redegøre for rekursionsligninger og renters renter. Samt hvordan man bruger og beregner rekursionsligninger og renters renter. Jeg vil også tale om brugen af programmering i matematik og omvendt, da det er en vigtig begrundelse for blandingen af disse fag til at starte med.

Teori:

Matematik:

Hvad er renters renter og rekursionsligninger?:

Renters renter er når du bliver ved med at lægge renter til et tal. Hvad jeg mener med dette er at hvis man har 100kr. og bliver ved med at ligge renter til stiger antallet af penge man får, hver gang.

Dette kan ske eksponentielt fra f.eks. 100kr. til 110kr. og i sidste ende et kæmpestort tal. En rekursionsligning er en ligning som indeholder sig selv. For eksempel de 110 kr. vi har nu, er det samme som de 100kr. vi havde før med 10 procent oven i. Skrevet op på ligningsform som

$$x_n = x_{n-1} \cdot 1,1.$$

Beregning af renters renter med rekursionsligninger:

Vores rekursionsligning for udredning af renters renter lyder $x_n = x_{n-1} \cdot (1 + r)$ ud fra denne ligning vil vi gerne løse karakterligningen og finde vores generelle løsning. Først skal vi starte med at finde vores karakterligning. Vi ved først og fremmest at en homogen lineær rekursionsligning af

k værdi, hvor at $n = 1, 2, 3 \dots$ er $r^k - a_1 \cdot r^{k-1} - a_k \cdot r^{k-k} = 0$ den kan man forsætte jo større ens k værdi er. Men vi har at gøre med en rekursionsligning af første orden altså er vores k lig med 1.

Derfor skal vi kun bruge den første del af ligningen $r^k - a_1 \cdot r^{k-1} = 0$ og indsætte vores k og a_1 værdi. a_1 er for os (1+re). Når vi har sat dem ind, kan vi solve for r. som giver os svaret $r = 1 + re$

$$r^1 - (1 + re) \cdot r^{1-1} = 0 \xrightarrow{\text{solve for r}} [r = 1 + re]$$

Vi kan ud fra dette opstille en ligning $x_n = (1 + r)^n$. Hvad det vil sige er at tager vores rente og opløfter n. Her står n for "antal år" hvilket vil sige at vi kan regne renten ud for bestemte år ved at udskifte n. Vi har dog ikke nogen begyndelses værdi og kan derfor ikke regne renters renter ud, bare med karakterligningen. Den ligning vi gerne vil finde frem til, skal ligne den her

$x_n = b \cdot (1 + r)^n$. Her har vi b som begyndelses værdi som bliver ganget med renterne og opløftet i antal år.

For at komme frem til $x_n = b \cdot (1 + r)^n$ skal vi bruge den generelle løsning. Det gælder at en homogen lineær rekursionsligning af k'te orden har den generelle løsning $x_n = b_1 \cdot r_1^n + b_2 \cdot r_2^n + b_k \cdot r_k^n$. Hvad det vil sige er at lige meget hvor stor ens orden er kan man bruge formlen. Vi har det dog en del nemmere da vi har en homogen lineær rekursionsligning af første orden og skal derfor kun bruge $x_n = b \cdot r_1^n$ vores r værdi kender vi allerede da vi fandt den med karakterligningen. Vi kan sætte vores r værdi ind på r_1 's plads og få $x_n = b \cdot (1 + r)^n$. Nu har vi fundet vores ligning til at regne renters renter. b er vores begyndelses værdi, for eksempel 1000kr. som man sætter ind i banken. Vi har så r som er renten hvilket kunne være 0,05 altså 5% i rente. 0,05 bliver lagt sammen med 1 for at give 1,05, som vi kan gange med begyndelsesværdien for at få 5% oveni den. Og n er hvor mange år pengene ligger i banken lad os sige de ligger 7 år i banken så får vi $x_n = 1000 \cdot (1 + 0,05)^7 \approx 1407,1$ vi kan se at vi så ville have 1407,1 kr. i banken.

Talfølger:

Forståelsen af rekursionsligninger kan forøges ved brugen af talfølger. Kort forklaret er talfølger uendelig mange tal, men skrevet i en bestemt rækkefølge. Ting som vi lærte i 1 klasse så som 2 tabellen og 7 tabellen er begge talfølger. Men i rekursionligningernes tilfælde beskriver talfølgen vores rekursionsligning. Lad os tage udgangspunkt i rekursionsligningen vi havde før, nemlig $x_n = x_{n-1} \cdot (1 + r)$. Hvis n er antal år, vil vores talfølge beskrive hvad x er hvert år. For at visualisere dette har jeg sat det hele ind i Excel. Vi tager udgangspunkt i at vores begyndelses værdi er 100 og at vores rente (r) er 5%

n	x
0	100,0
1	105,0
2	110,3
3	115,8
4	121,6
5	127,6
6	134,0
7	140,7
8	147,7
9	155,1
10	162,9

Det vi ser her, er dele af talfølgen. Vi ser nemlig kun mellem 0 og 10, mens at den i virkeligheden fortsætter uendeligt. Når vi bruger en talfølge, kan vi lettere se hvor meget den vokser hvert år og se en eksponentiel stigning i penge. Normalt ville vi skulle indtaste hvor mange år frem vi vil, mens at vi med talfølgen kan finde alle år med det samme.

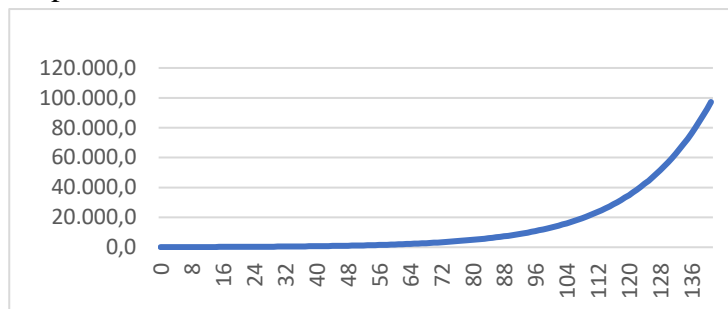
En af de mest kendte talfølger er Fibonacci talfølgen. Fibonacci talfølger beskriver Fibonacci spiralen som minder om den gyldne spiral som der findes på for eksempel snegles sneglehus. Denne talfølge kan først og fremmest findes som rekursionsligning $x_n = x_{n-1} + x_{n-2}$, hvilket oversat betyder at det tal vi har nu, er det forrige tal lagt sammen med tallet før det forrige. Det giver os talfølgen

n	x
0	0
1	1
2	1
3	2
4	3
5	5
6	8
7	13
8	21
9	34
10	55
11	89
12	144
13	233
14	377
15	610
16	987
17	1597
18	2584
19	4181
20	6765
21	10946
22	17711
23	28657
24	46368
25	75025

Vi kan ved hvert trin se at det er trinnet før lagt sammen med trinnet 2 gange før. Så som n_{11} som er 89 findes ved at lægge $n_{10} + n_9$ sammen altså $55 + 34 = 89$ Hvilket passer med hvad n_{11} er. Det er også en måde at dobbelt tjekke at man har lavet ens rekursionsligning korrekt.

Ekspontielle funktioner:

Når vi taler om rekursionsligninger, renters renter og talfølger kan vi ikke undgå at tale om eksponentielle funktioner. En eksponentiel funktion er en funktion der stiger mere og mere jo længere der går, altså stiger den eksponentielt. Vi har i vores renters renter regning at gøre med en eksponentiel funktion.



Jo flere år man går hen langs x-aksen jo mere stiger pengene op langs y-aksen. Dette er typisk sådan en eksponentiel funktion ser ud. Så i teorien kunne du sætte 100kr. ind i en bank og med 5% rente kunne du om 100 år stikke af med 13.150 tusinde kr.

Det skal dog siges at grafen kan være upræcis at se på, da det ligner at vi starter med et beløb på 0kr. men man kan selvfølgelig ikke have en eksponentiel funktion hvis man 0 kr. har.

Programmering i matematik:

I matematikkens verden bruger vi en del programmering til at gøre det hele lidt nemmere. 2 store eksempler på dette kan komme fra programmerne Maple og Excel. Når vi bruger Maple, bruger vi også næsten hele tiden kommandoer som er en form for funktion vi kalder efter. Et eksempel på dette kunne være når vi gerne vil bruge "Solve" kommandoen i Maple. I programmet er der en funktion de har kaldet "solve". De har så programmeret hvad der skal ske når funktionen bliver kaldt, hvilket er at løse en ligning eller regnestykke. Og bare i den ene funktion gemmer der sig masser af "if" statements og lign.

Hvis vi tager et kig på det populære program Excel kan vi også se hvor tit vi bruger programmering. For når vi sætter et regnestykke op der kan vi kalde på andre koloner eller celler. Igen er det som at kalde på en funktion som indeholder noget. I Excel kunne cellen A3 indeholde tallet 3. Så når vi kalder på A3 ved den at der står 3 i den celle. Det er en form for variabel som kan være alt fra tal til regnestykker. Alt i alt bruges der store mængder programmering i matematik. Hvad

programmerings indblanding i matematikkens verden har gjort, er at forøge brugervenlighed, tilgængelighed og forståelse.

Programmering:

Variabler:

I programmering bruges variabler ofte. En variabel kan give et navn eller tegn en vis værdi. Det kan blive brugt til at ændre værdier dynamisk og til at organisere ens kode. I JavaScript bruger man oftest "var" eller "let" til at deklarere variabler. På samme måde som i matematik hvor man giver "A" en værdi på 3, kan man i programmering gøre det samme.

```
let b = start.value();  
let y = rente.value();  
let n = aar.value();
```

I mit tilfælde bruger vi let til at definere 3 værdier. Disse 3 værdier er vores startværdi altså hvor mange penge man starter med at have inde i banken. Så har vi y som er vores rente i procent, så hvis man vil have en rente på 5% skal den være lig 5. Den sidste variabel er n som definerer hvor mange år efter startåret vi vil gå frem. Forskellen på var og let handler om hvorvidt de kan kaldes overalt i ens kode eller kun inden i en funktion for eksempel. Hvis du skriver, "var" i starten af ens kode kan man bruge den i resten af koden. Men hvis du skriver en "let" i starten, er det kun der den kan bruges. Dette hjælper med ikke at skabe konfliktende kode, da man ved hjælp fra "let" kan specificere dens værdi.

Funktioner:

Funktioner er endnu en af de mest vigtige kommandoer at kende til. Hvad funktioner gør er at man nemmere kan kalde visse linjer koder. For at gøre dette skal man lave en funktion med navn og indsætte koden indeni funktionen. Da funktionen har et navn, kan man kalde på den. Lidt ligesom en person kan kalde efter Mathilde og Agda, kan en computer kalde efter "calculate" for eksempel.

```
function calculate() {  
  background(220);  
  b = start.value();  
  y = rente.value();  
  n = aar.value();  
}
```

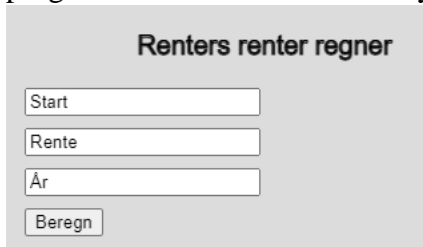
I mit tilfælde bliver funktionen brugt til at regne vores renters renter ud. Hvis man bare skrev koden til udregning, ville den blive udført automatisk. Hvad det betyder er at uden en funktion vil koden blive kørt før vi får sat vores tal ind. Og derved ville vi ikke få et svar. På den anden side kunne man skrive koden i "draw", hvis man gjorde dette, ville den kører hver frame og derved blive ved med at sende den samme besked og derved ødelægge det for brugeren. Men ved hjælp fra funktioner aktiverer vi kun koden når vi har lyst. Vores funktioner bliver kørt ved at trykke på en knap med musen. Som man kan se på billede, kalder den navnet "calculate" som var vores funktion.

```
button = createButton("Chonk");  
button.position(20,155);  
button.mousePressed(calculate);
```

Men det kan også aktiveres på en masse forskellige måder, ved at klikke på en knap på tastaturet eller efter noget tid er gået. Dette giver programmører en lang række måder at løse opgaver på. Vi har ved hjælp fra funktioner skabt en form for lommeregner der kan regne renters renter ud.

Input og Buttons:

Et input kan sørge for at man kan give en værdi når programmet kører. Vi kan derfor når vi tænder programmet skrive hvad vi har lyst til inde i boksen. Vi kan med informationen i boksen gøre en



Renters renter regner

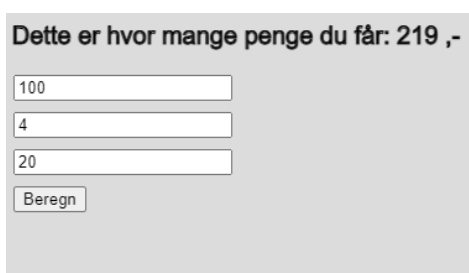
Start

Rente

År

Beregn

masse. Jeg bruger input til at beregne vores renters renter. Uden input skulle man ind i koden for at skifte for eksempel vores startværdi eller rente, men med inputs kan man skifte det nemt og hurtigt. For en person som ikke koder er det også nemmere at vide hvor man skal skrive renter ind henne. Alt i alt giver det en bedre brugerflade. Buttons bliver brugt i sammenhæng med vores inputs til at beregne renters renter ud, for når alle værdierne er skrevet ind, trykker vi på "button" og så fortæller vi programmet at den nu skal regne det ud for os. Dette simple brug af inputs og buttons gør vores program overskueligt og nemt at forstå.



Dette er hvor mange penge du får: 219 ,-

100

4

20

Beregn

Betinget udførelse:

Betinget udførelse er også et super vigtigt udtryk at kende til. Det vil hjælpe en med at aktivere dele af sin kode når en vis ting sker. I JavaScript vil man oftest støde på udtrykkende "if" og "else" groft oversat til hvis og ellers. For at vise et eksempel kunne man sige "hvis" variabelen a er det samme som 2 så skal der tegnes en cirkel "ellers" skal der tegnes et kryds. På denne måde kan man kontrollere udkommet af en handling i sit program.

```
if (isNaN(start.value()) || isNaN(rente.value()) || isNaN(aar.value())) {  
    console.log("please use numbers");  
}
```

Her siger vi for eksempel at hvis vores variabler b, y og n ikke består af tal skal den bede brugeren om kun at sætte tal i input felterne. Vi kan nemlig ikke regne renters renter hvis vi ingen tal har at arbejde med. Dog skal vi jo, regne renters renter ud hvis der står tal i input. For at gøre dette så nemt som muligt bruger vi "else" kommandoen. "Else" kommandoen kan kun blive brugt sammen med en "if" kommando, for man kan ikke sige "ellers" hvis man ikke har et "hvis".

Så hvis man har tastet noget som ikke er et tal i input, så spørger programmet en om at bruge tal og hvis den ikke spørger en om at bruge tal, så betyder det jo at alle input har tal. Derfor aktiveres koden som under "else" og renters renter bliver udregnet.

```
else  
    text("Dette er hvor mange penge du får: " + Math.floor(b * Math.pow((1+(y/100)),n)) + " , -  
    ",width/2,40);  
    console.log(Math.floor(b * Math.pow((1+(y/100)),n)));
```

Matematik i programmering:

Matematik og programmering går hånd i hånd som nat og dag. Vi bruger faktisk næsten hele tiden matematik i programmering. Det kan være alt fra at få en firkant til at gå 1 pixel til venstre hver frame, til at lave en lommeregner. Vi kender også variabler fra matematikkens verden, disse variabler blandet med simpel matematik kan skabe en helt ny verden på ens skærm. Hvis vi forestiller os vi har en cirkel med centrum i c. c er en variabel som vi lige nu kan sætte i midten af skærmen, det vil sige at hvis vi har en 400 gange 400 skærm er c i 200 hen af x akse og 200 ned langs y akse. Allerede nu har vi brugt matematik til at skabe en cirkel, men ville det ikke være federe hvis vi kunne få den til at bevæge sig? Vi kan med simple ting vi kender fra matematik få det til at ske, vi trækker bare 1 fra c's x akse. Så hvis c's x koordinat hedder x kan vi bare sige "x-1" eller "x--". Hvis det er blevet skrevet inde under "draw funktionen" vil cirklen nu hvert

sekund rykke 1 pixel til venstre, altså har vi brugt matematik til at animere et objekt i programmering.

Også vores renters renter lommeregner bruger matematik. Ikke nok med at bruge et regnestykke for renters rente altså $b \cdot \left(1 + \frac{y}{100}\right)^n$, men også matematiske udtryk som floor som bliver brugt til at runde ned til nærmeste heltal.

```
text("Dette er hvor mange penge du får: " + Math.floor(b * Math.pow((1+(y/100)),n)) + " , -",width/2,40);  
console.log(Math.floor(b * Math.pow((1+(y/100)),n)));
```

Noget så simpelt som dette kræver masser af matematik, og ud fra dette kan vi også konkludere at der ikke ville findes programmering uden matematik.

Vurdering:

Hvordan vurderer jeg renters renter og rekursionsligninger via matematik og programmering? Vi kan i teorien ikke adskille disse to da der altid vil indgå matematik i programmering og omvendt. Men hvis vi kigger på måden det er blevet beregnet i matematik afsnittet versus programmerings afsnittet hvilken er så bedst? Jeg syntes selv at bruge den matematiske metode er den bedste og nemmeste. Dette er grundet af at programmet i programmering kun kan regne noget meget bestemt, den kan kun finde et år ad gangen. Ved hjælp fra matematik kan man langt nemmere finde frem til et bestemt år, men også lave en graf på langt flere år. Talrækker er også langt nemmere at beregne med det matematiske værktøj Excel. Dog kan det argumenteres at der indgår programmering i opsætningen af Excel ark der kan regne talfølger. Det skal dog siges at man kunne gøre løsningen af renters renter og rekursionsligninger nemmere med programmering, men dette er kun hvis man bruger mange timer på at opsætte forskellige programmer der kan løse hver eneste ting. Hvis man havde tiden, ville programmering vinde stort, ligesom at nogle har programmeret Maple og Excel som har gjort matematik nemmere. Kunne man også selv lave et program der gør det endnu nemmere.

Konklusion:

Hvad er renters renter og hvordan regnes det i fagene?

Renters renter er fænomenet når renter bliver lagt til et beløb igen og igen. Beløbet vil derfor blive ved med at stige eksponentielt og ikke lineært. I matematik bruger man rekursionsligninger til at beregne renters renter, ved hjælp af rekursionsligninger kan man nemt og hurtigt beregne renters renter. I programmering bruger vi også rekursionsligninger til at beregne renters renter, dog kan vi i programmering lave et program ved hjælp af diverse kommandoer, som endnu hurtigere kan regne renters renter ud for os. Man kan på den måde sige at programmering kan fremskynde den matematiske proces.

Hvad er rekursionsligninger?

En rekursionsligning er en ligning som refererer til sig selv. Den indeholder altså sig selv og kan derfor beregne hele talfølger og i vores tilfælde renters renter. Et eksempel kunne være $x_n = x_{n-1} \cdot 1.1$. Her siger vi at x_n er det samme som det x_{n-1} vi havde før ganget med 1.1.

Hvordan hænger rekursionsligninger og renters renter sammen?

Renters renter er en rekursionsligning da den indeholder sig selv. For de penge vi har nu, er det samme som de penge vi havde før med rente oveni $x_n = x_{n-1} \cdot \text{rente}$. Derfor kan vi ikke tale om renters renter uden rekursionsligninger.

Hvordan kan man bruge programmering som et værktøj til at løse renters renter?

Programmering er et problemløsningsfag. Du har et problem eller en ide som skal laves, så kan du gøre det med programmering. Du kan i teorien lave lige hvad du har lyst til med programmering som i vores tilfælde var at løse renters renter nemmere og hurtigere. Vi ville også gerne gøre det til et program som selv folk der havde det svært ved renters renter og matematik at bruge.

Hvordan hænger programmering og matematik sammen?

Jeg har med denne opgave gerne finde ud af hvorfor det er at programmering og matematik hænger sammen. Jeg har taget et kig på programmering i matematikkens verden og omvendt. Og ud fra dette har jeg konkluderet at den ene ikke ville kunne findes uden den anden, i hvert fald i nutiden. Før i tiden kunne man godt have matematik uden at have programmering da man skulle gøre alt i hånden. Men op i nyere tid foregår meget matematik via kommandoer, disse kommandoer er noget

som programmering har masser af. Og omvendt ville vi ikke kunne programmere uden matematik da matematik er grunden til at det hele fungerer.

Litteraturliste og kilder:

<https://mathtxa.systime.dk/?id=505> (systime Mat A HTX)

Dokumenter (google docs) delt med os af vores programmerings lærer Henrik

<https://mathtxa.systime.dk/index.php?id=508#c2556> (Fibonacci talfølgen)

Bilag:

```
1▼ function setup() {  
2  
3   createCanvas(400, 400);  
4   background(220);  
5  
6   stroke(0);  
7   textSize(20);  
8   textAlign(CENTER);  
9   text("Renters renter regner", width/2, 40);  
10  
11  start = createInput("Start");  
12  start.position(20, 65);  
13  
14  rente = createInput("Rente");  
15  rente.position(20, 95);  
16  
17  aar = createInput("År");  
18  aar.position(20, 125);  
19  
20  let b = start.value();  
21  let y = rente.value();  
22  let n = aar.value();  
23  
24  button = createButton("Beregn");  
25  button.position(20, 155);  
26  button.mousePressed(calculate);  
27  
28  
29 }
```

```
30
31 ▼ function calculate() {
32   background(220);
33   b = start.value();
34   y = rente.value();
35   n = aar.value();
36
37 ▼   if (isNaN(start.value()) || isNaN(rente.value()) || isNaN(aar.value())) {
38     console.log("please use numbers");
39
40     text("Please use numbers", width/2,40);
41     start.value("Start");
42     rente.value("Rente");
43     aar.value("År");
44   }
45
46   else
47
48     text("Dette er hvor mange penge du får: " + Math.floor(b * Math.pow((1+(y/100)),n)) + " , -",width/2,40);
49     console.log(Math.floor(b * Math.pow((1+(y/100)),n)));
50 }
51
52 ▼ function draw() {
53
54 }
```