TMA4101 – oblig

Matte problemer og slikt

Werner Kristoffer Korbi – 11/17/2024

Sammendrag:

I denne rapporten vil det bli undersøkt diverse matteproblemer, og de blir demonstrert for enkel forfatning. Det vil også bli demonstrert diverse programmer som ble laget for TDT4100, for matte problemene eller for gøy, som var artige å holde på med.

1 Teori

I del 1 av rapporten blir et problem fra 1937 utforsket. Problemet heter «The Collatz conjencture», og ble introdusert av tysk matematiker Lothar Collatz. Problemet spør om to enkle aritmetiske operasjoner kan gjøre hvilke som helst tall til tallet 1. Problemet omhandler en sekvens av heltall, hvor alle nye tall er avledet fra det forrige. Hvis tallet er et partall skal tallet deles på to, hvis tallet er oddetall skal tallet bli ganget med 3, også plusset med 1. Problemet har blitt sett til å virke på tall så store som 10^{20} og større, men det er ikke blitt vist noe bevis på hvorfor det virker. (Collatz conjecture, Wikipedia.)

I del 2 av rapporten skal en algoritme undersøkes, denne kalles ofte 196-algoritmen. Algoritmen går slik; ta et nummer, reverser rekkefølgen, (f.eks. 201 blir 102), så summerer man det originale nummeret med det reverserte nummeret. For nesten alle nummer under 10 000 som går gjennom denne prosessen, blir de til et palindrom, altså er tallet likt baklengs som forlengs, (f.eks. 56 blir til 121), de fleste av disse tallene når palindrom-status innen 4 iterasjoner, med noen unntak. Noen nummer blir ikke til palindromer, disse kalles Lychrel-nummer, (Lychrel kommer fra et anagram av navnet Cheryl, som var førstenavnet til kjæresten til Wade Van Landingham, som navnga disse tallene, det er romantisk det). Problemet her er å bevise at 196 er det minste lychrel-nummeret. Selv hvor mange iterasjoner problemet kan bli kjørt gjennom, er det ikke nok bevis for å bestemt si at nummeret er et lychrel-nummer. (Lychrel number, Wikipedia), (Steve Hurley, 2017), (Weisstein, Eric W. Wolfram MathWorld)

I del 3 av rapporten blir det skrevet om noen artige programmer som ble laget for, som sagt, TDT4100.

2 Eksperimentelt + Resultater + diskusjon

2.1 The Collatz Conjecture

I dette forsøket ble et enkelt program laget, vedlegg (1.1), dette programmet viser hvordan enhver tilfeldig verdi fra 1 – 100 blir til slutt 1. 100 er et vilkårlig valg av en grense, teoretisk sett finnes det muligvis ikke en grense. Det er heller ikke mulighet for dette systemet å prosessere et for stort tall, da hadde PC-en krasjet. Dette programmet kan også skrives om som en funksjon.

$$f(n)\begin{cases}\frac{n}{2} & if\ n = 0\ (mod 2)\\ 3n+1 & if\ n = 1\ (mod 2)\end{cases} \quad (Collatz\ conjecture, Wikipedia)$$

Hvor mod er Pythons «%» funksjon.

Det høyeste tallet plottet i Collatz Conjecture er 2^{100000}-1, (W. Ren, S. Li, R. Xiao, W. Bi), dette tallet tok 1,344,926 iterasjoner for å nå 1, og en mye kraftigere datamaskin enn denne IdeaPad Slim 3 14AMN8 har kapabiliteten til å være.

Det er mye mer kompliserte greier i dette matteproblemet, som ikke er lett å forstå. Matematiker Paul Erdös, en av de mest betydningsfulle matematikerne i historien, ved siden av folk som Euler, sa til å med; «Mathematics may not be ready for such problems.», og til den dag i dag, er det fortsatt ikke helt bevis eller motbevis mot denne formodningen, men mange bevis har blitt produsert for å støtte dette. F.eks. beviste Krasikov og Lagaris, med hjelp av et datasystem, at for et intervall [1, x], så må antall heltall som når 1 være i hvert fall lik $x^{0,84}$ (Krasikov og Lagaris, 2003). I rett tid vil nok dette ble bevist eller motbevist, men ikke i dag, sannsynligvis.

2.2 196-problemet

«Reverse then add» - algoritmen er som sagt en algoritme som reverserer nummeret, så adderer det til opprinnelige nummeret, også gjør det det om igjen og om igjen. I denne situasjonen og programmet i vedlegg 2.1, så stopper den når produktet er et palindrom. To eksempler på kjøring av programmet er i vedlegg 2.2. Som algoritmen forutser, så vil de fleste tall under 10 000 i programmet bli et palindrom innen 4 steg. Det er et enkelt program, som viser et kult problem.

Som også visst i vedlegg 2.2, så satte jeg nummeret til å være 196, og den stakkars pc-en klarte å holde seg gående overraskende lenge, programmet ble bare virkelig tregt. Etter 2 minutter og 9 sekunder krasjet hele tingen. Da programmet ble åpnet etter krasjet ble den siste verdien funnet, den står som sagt i vedlegg 2.2. Dette betyr at hvis 196 blir et palindrom, skjer det etter den verdien, da må man ha en ganske kraftig pc. I 1987 til 1990 programmerte John Walker et program for akkurat dette, det kjørte da i 3 år når pc-en hans var på, det siste resultatet var at tallet ikke ble et palindrom, men nådde 1 million siffer, og gikk gjennom 2 415 836 iterasjoner. En Jason Doucette fortsatte søket etter palindromet og nådde 12,5 millioner siffer i mai 2000. Etter det har Wade Van Landigham fortsatt, og i 2006 nådde han 300 millioner siffer. I 2011 klarte Romain Dolbeau å nå 414 millioner siffer etter en billion iterasjoner. I 2015 nådde han en billion siffer. Det viser bare hvor avansert datamaskiner har blitt, og hvor langt dette palindromet må være hvis det noen gang skal oppdages.

Men dessverre kan det aldri bevises med garanti om tallet er et palindrom, det kan bare motbevises, for programmet kan ikke kjøre gjennom alle mulige tall, det hadde tatt for lang tid og en for kraftig pc. Men en artig jakt på nummer uansett.

2.3 program

Dette er ikke et veldig intensivt delkapittel, det ble laget noen artige programmer, f.eks. I vedlegg 3.1 ble det laget et tre på rad spill, som var moro. Det hadde nok vært mulig å lage et mye mer optimalisert spill, men dette har en plass i hjertet til kodeskriveren. Deretter i vedlegg 3.2, blir et wordle spill laget, som ikke lager ekte ord, så det ødelegger litt hensikten med spillet, men en dag skal det bli fikset. For nå kalles det wrodle, siden det ikke er ekte ord.

3 Kildehenvisning

Collatz conjecture, Wikipedia, The Free Encyclopedia, Wikimedia Foundation, 14.10.2024, https://en.wikipedia.org/wiki/Collatz_conjecture

Weisstein, Eric W. "196-Algorithm." From *MathWorld*--A Wolfram Web Resource. https://mathworld.wolfram.com/196-Algorithm.html

Steve Hurley, (2017), *196-an unsolved problem*. Tilgjengelig fra (https://explainingscience.org/2017/08/12/196-an-unsolved-problem/)

Lychrel number, Wikipedia, The Free Encyclopedia, Wikimedia Foundation, 22.08.2024, https://en.wikipedia.org/wiki/Lychrel_number

W. Ren, S. Li, R. Xiao and W. Bi, "Collatz Conjecture for 2^100000-1 Is True - Algorithms for Verifying Extremely Large Numbers," 2018 IEEE SmartWorld, https://ieeexplore.ieee.org/document/8560077

*Krasikov, Ilia; Lagarias, Jeffrey C. (2003). "Bounds for the 3x + 1 problem using difference inequalities". Acta Arithmetica. **109** (3): 237–258*

4 Vedlegg

Vedlegg 1.1 The Collatz Conjecture program

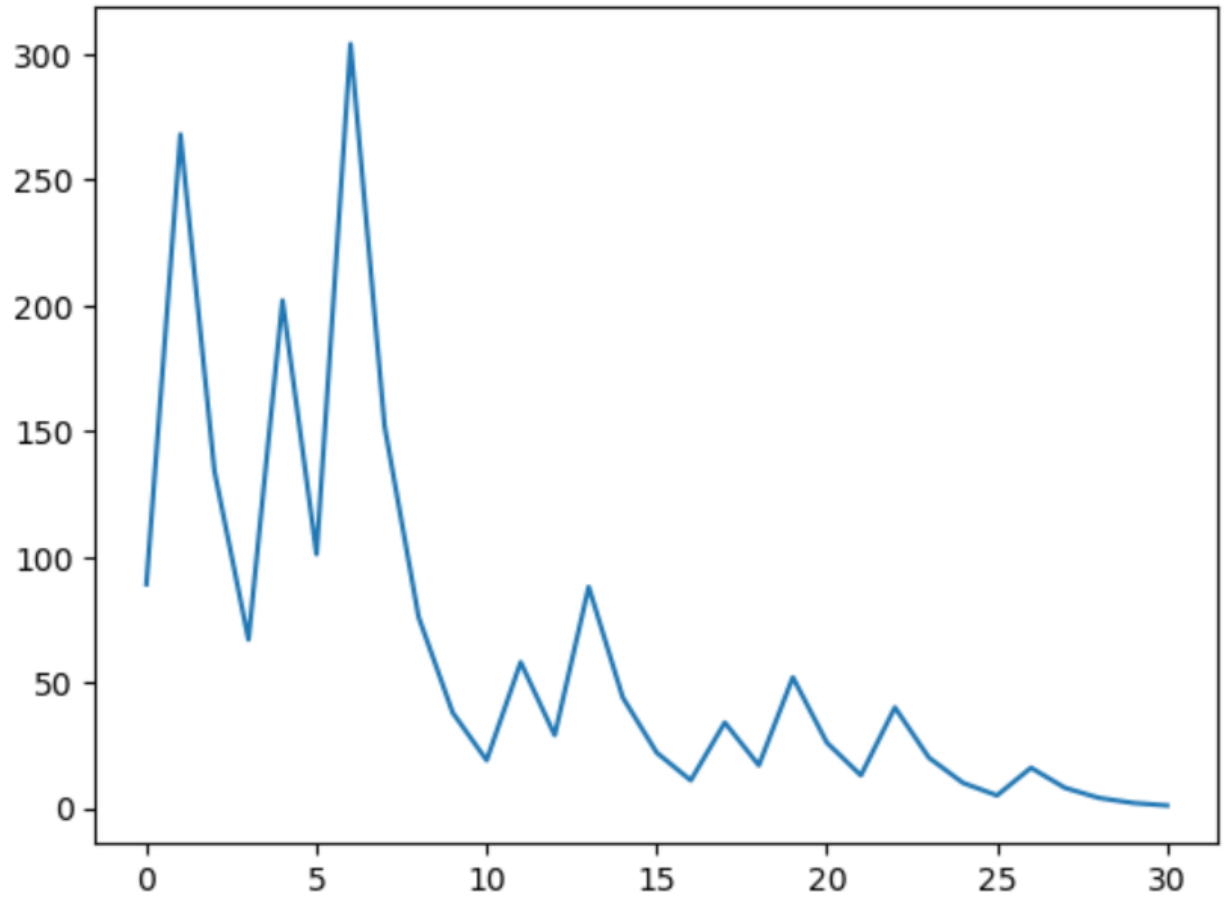
Her ble det laget et kort program for å demonstrere problemet:

```
 1. import random
 2. import matplotlib.pyplot as plt
 3. x = random.randint(1, 100)
 4.
 5. list = [x]
 6.
 7. print(x)
 8.
 9. while x > 1:
10.     if x%2!=0:
11.         x = 3*x+1
```

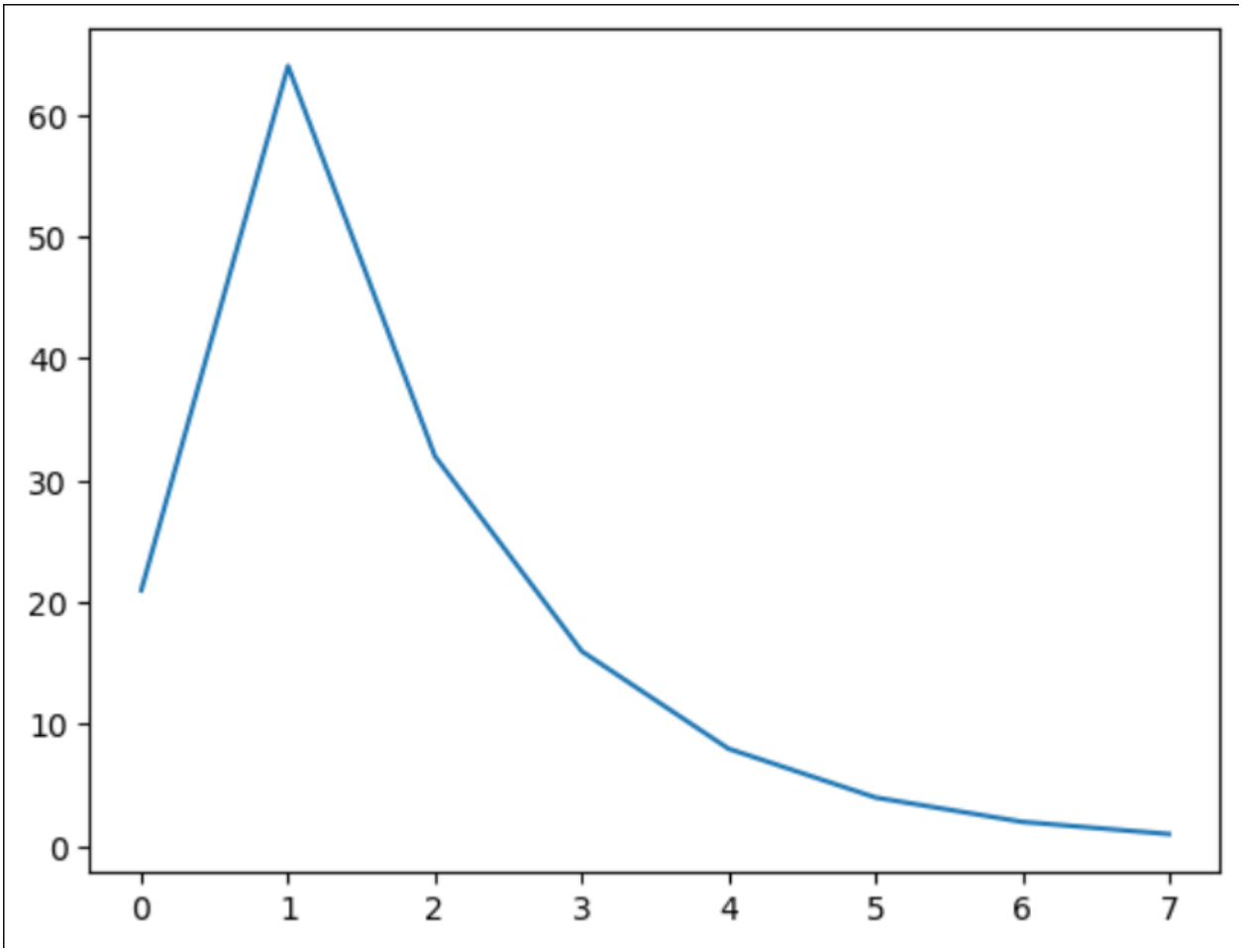
```
        print(x)
        list.append(x)
    elif x%2==0:
        x = x//2
        print(x)
        list.append(x)

list2 = []
for i in range(len(list)):
    list2.append(i)

plt.plot(list2, list)
```

Vedlegg 1.2 Eksempel på kjøringer av program i vedlegg 1.1

Her er grafen av tall når tallet 89 plottes inn:



Her er grafen når 21 blir plottet inn:



Vedlegg 2.1 program for 196-algoritmen

Et enkel Python program som printer ut riktig palindrom verdi for alle verdier testet

```
org_num = 56
num2 = org_num
reversed_num = 0
true = False
while true != True:
    num = num2 + reversed_num
    num2 = num
    reversed_num = 0
    while num != 0:
        digit = num % 10
        reversed_num = reversed_num * 10 + digit
        num //= 10
    print(num2)
    if reversed_num == num2:
        print(num2)
        break
```

Vedlegg 2.2 eksempler på kjøring

Her har det blitt satt inn to forskjellige verdier, 56, som har en veldig kort kjøretid, og tilfeldig plukket 18586, som har litt lenger kjøretid

```
[6]: org_num = 56
num2 = org_num
reversed_num = 0
true = False
while true != True:
    num = num2 + reversed_num
    num2 = num
    reversed_num = 0
    while num != 0:
        digit = num % 10
        reversed_num = reversed_num * 10 + digit
        num //= 10
    print(num2)
    if reversed_num == num2:
        true = True
        print(num2)
        break
```

```
56
121
121
```

```
14]: org_num = 18586
num2 = org_num
reversed_num = 0
true = False
while true != True:
    num = num2 + reversed_num
    num2 = num
    reversed_num = 0
    while num != 0:
        digit = num % 10
        reversed_num = reversed_num * 10 + digit
        num //= 10
    print(num2)
    if reversed_num == num2:
        true = True
        print(num2)
        break
```

```
18586
87167
163345
706706
1314313
4448444
4448444
```

Så prøvde jeg å sette inn 196, deretter ble programmet spinngal, og sluttet aldri (som forventet), dette er det siste tallet programmet fant da jeg sjekket etter krasjet.

```
466912970847568791834137040717835767533627277251018654286485094451423800314172974626913526386894379786929085852801224171777295971683983558792946275130876329364499996907101327451953291819862686034727876555676293673214382406243003531493506630168409253329369757256020574118764651198638836634314766791339541602938495690536383574665436059591032184709267545218407271319358365702612720309032901638018922989244373496231608903327861062449679823322357955005324010955799467225546106457499281683499634158324575421644325290342606572100347505548447796668389980404943994077539948313164013373566562924706236590963207821740554177564504452911140988850720601340475334154050682202909906200595075253384861451163159839298726449880901173272946959699520583526271676309300645181428547914067058997685496113115665535090725952403976903024282920948887242419196996497166040702376122711470960997537971530050337634438530193673458932541114527230724105903624115539628911121431900431474029103347130330233421301324927704938601244736647264031400290521014622767821991119583019212724025162906752587979558031380521413101594824344731531641873453396676139684952114858792392090637325637354729928859130751413620284113474880317826322310927497322636514344817461977102451059869521288168678707649609961041461408335998836030707068892592133495301822629550112800499767995098311054837217202605331186288870716920747800632804264140159006056696876960883115968961144301769164717453425625223794829113222628803178464410492927304148130858829928552636523845991183308847521248487041676583355369246026226443429486201314124974131854879785257709261530416312020276021090228767226410124993004020462856626442106840496828533103014441923413174330083038403509923412110893583552132739959053692283541114522896437628202594442584295092527884470006006520732167319713956278469970192424168895892027252931957040415061708064556651131169349768995967140874593408254600291367607361439591599706963828326111007905461690283896225214416849435148159500160901920217704045154248493310712694889014110935439446577145613612979137008563270653816566647331046021395894468039944840507999376669774484550574409028560524310242344721456542396033699438718290575459075552276390746010943250946085322233279858442711578323089062325853744418903189198460982409031172072965748528142717137125457729074811410860595345664852946349974849383161449331887565234257389258921464679214750195528668738233629038610367152851253102427042844122673826765556787285305871590181933481657231007087000053540336680225626502879652803772794838770724220983475908306779744886946143207154803605229983142643915736934558200427728363346675497071397403391878657490703086641
```

Vedlegg 3.1 Tre på rad

Det ble først definert flere funksjoner for å hjelpe til i spillet

```
def lag_brett():
    print("      1   2   3   ")
    print("   --------------")
    print(f"1  |{pos1_1}  ||{pos1_2}  ||{pos1_3}  |")
    print("   --------------")
    print(f"2  |{pos2_1}  ||{pos2_2}  ||{pos2_3}  |")
    print("   --------------")
    print(f"3  |{pos3_1}  ||{pos3_2}  ||{pos3_3}  |")
    print("   --------------")

lag_brett()
```

```
def match_over():
    if ((pos1_1 == pos1_2 == pos1_3) or (pos1_1 == pos2_1 == pos3_1) or (pos1_1 == pos2_2 == pos3_3)) and pos1_1 != (" "):
        if pos1_1 == "x":
            return True
        elif pos1_1 == "o":
            return True
    elif ((pos1_2 == pos2_2 == pos3_2) or (pos2_1 == pos2_2 == pos2_3)) and pos2_2 != (" "):
        if pos2_2 == "x":
            return True
        elif pos2_2 == "o":
            return True
    elif ((pos3_1 == pos3_2 == pos3_3) or (pos1_3 == pos2_3 == pos3_3)) and pos3_3 != (" "):
        if pos3_3 == "x":
            return True
        elif pos3_3 == "o":
            return True
```

```
    elif (pos3_1 == pos2_2 == pos1_3) and pos3_1 != (" "):
        if pos3_1 == "x":
            return True
        elif pos3_1 == "o":
            return True
    else:
        return False
```

```
def navn():
    spiller1 = str(input("skriv inn navnet til spiller 1"))
    spiller2 = str(input("skriv inn navnet til spiller 2"))
    return spiller1, spiller2
```

```
def lovlig(move):
    if move == "pos1_1":
        if pos1_1 == " ":
            return True
        else:
            return False
    elif move == "pos1_2":
        if pos1_2 == " ":
            return True
        else:
            return False
    elif move == "pos1_3":
        if pos1_3 == " ":
            return True
        else:
            return False
    elif move == "pos2_1":
        if pos2_1 == " ":
            return True
        else:
            return False
    elif move == "pos2_2":
        if pos2_2 == " ":
            return True
        else:
            return False
    elif move == "pos2_3":
        if pos2_3 == " ":
            return True
        else:
            return False
    elif move == "pos3_1":
        if pos3_1 == " ":
            return True
        else:
            return False
    elif move == "pos3_2":
        if pos3_2 == " ":
            return True
        else:
            return False
    elif move == "pos3_3":
        if pos3_3 == " ":
            return True
        else:
            return False

```

```
1. def riktig(move):
2.     if move == "pos1_1" or "pos1_2" or "pos1_3" or "pos2_1" or "pos2_2" or "pos2_3" or "pos3_1"
or "pos3_2" or "pos3_3":
3.         return True
4.     else:
5.         return False
6.
```

Så selve spillet:

```
 1. pos1_1 = " "
 2. pos1_2 = " "
 3. pos1_3 = " "
 4. pos2_1 = " "
 5. pos2_2 = " "
 6. pos2_3 = " "
 7. pos3_1 = " "
 8. pos3_2 = " "
 9. pos3_3 = " "
10.
11. lag_brett()
12. spiller1, spiller2 = navn()
13. match_going=True
14.
15. while match_going == True:
16.     spiller1_tur = True
17.     spiller2_tur = True
18.     while spiller1_tur == True:
19.         move = str(input("Spiller 1 sin tur"))
20.         if riktig(move) == True:
21.             if lovlig(move) == True:
22.                 spiller1_tur = False
23.                 if move == "pos1_1":
24.                     pos1_1 = "x"
25.                 elif move == "pos1_2":
26.                     pos1_2 = "x"
27.                 elif move == "pos1_3":
28.                     pos1_3 = "x"
29.                 elif move == "pos2_1":
30.                     pos2_1 = "x"
31.                 elif move == "pos2_2":
32.                     pos2_2 = "x"
33.                 elif move == "pos2_3":
34.                     pos2_3 = "x"
35.                 elif move == "pos3_1":
36.                     pos3_1 = "x"
37.                 elif move == "pos3_2":
38.                     pos3_2 = "x"
39.                 elif move == "pos3_3":
40.                     pos3_3 = "x"
41.             else:
42.                 print("ulovlig")
43.         else:
44.             print("Feil navn på turen")
45.     if match_over()==True:
46.         print("Spiller 1 har vunnet!!!")
47.         match_going=False
48.         break
49.     if spiller1_tur == False:
50.         print(lag_brett())
51.
52.
```

```
53.
54.
55.     while spiller2_tur == True:
56.         move = str(input("Spiller 2 sin tur"))
57.         if riktig(move) == True:
58.             if lovlig(move) == True:
59.                 spiller2_tur = False
60.                 if move == "pos1_1":
61.                     pos1_1 = "o"
62.                 elif move == "pos1_2":
63.                     pos1_2 = "o"
64.                 elif move == "pos1_3":
65.                     pos1_3 = "o"
66.                 elif move == "pos2_1":
67.                     pos2_1 = "o"
68.                 elif move == "pos2_2":
69.                     pos2_2 = "o"
70.                 elif move == "pos2_3":
71.                     pos2_3 = "o"
72.                 elif move == "pos3_1":
73.                     pos3_1 = "o"
74.                 elif move == "pos3_2":
75.                     pos3_2 = "o"
76.                 elif move == "pos3_3":
77.                     pos3_3 = "o"
78.             else:
79.                 print("ulovlig")
80.         else:
81.             print("feil navn på turen")
82.     if match_over()==True:
83.         print("Spiller 2 har vunnet!!!")
84.         match_going=False
85.         break
86.     if spiller2_tur == False:
87.         print(lag_brett())
88.
```

Jeg synes i hvert fall det var imponerende da jeg skrev den.

Vedlegg 3.2

Her ble det laget et worlde program, som lagde tilfeldige ord, så ikke faktiske ord, den funksjonen klarte jeg ikke å lage.

```
1. import random
2. import numpy as np
3. dict = {1:"A", 2:"B", 3:"C", 4:"D", 5:"E", 6:"F", 7:"G", 8:"H", 9:"I", 10:"J", 11:"K", 12:"L",
13:"M", 14:"N", 15:"O", 16:"P", 17:"Q", 18:"R", 19:"S", 20:"T", 21:"U", 22:"V", 23:"W", 24:"X",
25:"Y", 26:"Z"}
4. ordet = []
5. for i in range(5):
6.     ordet.append(random.randint(1,26,))
7. n = 0
8. print(ordet)
9. wordet = []
```

```
for part in ordet:
    wordet.append(dict[part])
    print(wordet)

rikord = ["X", "X", "X", "X", "X"]
game_lost = False
game_won = False
losing_condition = 0
while game_lost == False and game_won == False:
    guess = str(input("Skriv gjettet her")).upper()
    if len(guess) == 5 and guess.isalpha() == True:
        for w in range(len(wordet)):
            if wordet[w] == guess[w]:
                rikord[w] = guess[w]
            else:
                pass
        print(rikord)
        losing_condition += 1
        if rikord == wordet:
            print("vinner")
            break
        if losing_condition == 5:
            print("taper")
            break


    else:
        print("invalid guess")

```