# Manus

**Side 1 Intro**

[J] Jeg er Jon-Fredrik og dette er Lars.

[L] Vi vil presentere vår simulering av Rossumøya

**Side 2 Struktur**

[L] her ser dere strukturen av klassene i programmet; Biosim. Biosim tar for seg visualiseringa og simuleringa, men for å gjøre dette må det få informasjon fra de andre funksjonene. Bynner på Island, som er selve øya, som består av flere celler i en matrise. Hver av disse cellene består av en type landskap. Island tar også for seg å gå gjennom alt som skal skje et år. Hvert landskap er enten jungle, savannah, desert, mountain eller ocean. Som man ser på strukturen er mountain og ocean en en type klasse som ikke har noen metoder eller variabler. Der det skjer ting er i jungle, desert og savannah. Jungle inneholder de fleste metodene og savannah og desert er underklasse av jungle, som da arver de fleste metodene bortsett fra mat-delen. Disse landskapene må igjen hente informasjon fra alle dyrene som er der, via Animals, som består av herbivores og carnivores, hvor carnivores er en underklasse av herbivores.

[J] Vi har organisert det slik at informasjonen bare flyter en vei. Island snakker ikke med dyra. Det som kan beregnes i dyra, beregnes der, så sendes informasjonen til landskapet. I landskapene bor dyra i lister. Landskapet håndterer denne lista med dyr, og får alle dyra til å spise,og dø osv. Øya styrer igjen landskapene, og leverer informasjonen til klassen BioSim for visualisering.

**Side 4 Own time [J]**

Her ser vi at det er numpy sin innebygde choice metode som tar lengst tid. Den brukes til å bestemme om dyra skal beverge seg til høyre, venstre, opp eller ned. Vi lurer på om den kunne vært raskere om vi hadde gitt den array, for nå jobber den med en liste. Vi kunne også prøvde å lage vår egen metode, men vi trodde vel at numpy skulle være rask og enkel å bruke.

Carnivore.feeding tar tid, men vi finner ingen lavt hengende frukt. Det er bare mange til som skjer der. Det er mange byttedyr som skal sjekkes, og flere situasjoner utifra fitnessen til dyra seg i mellom.

Vi tror utdate\_fitness kunne vært gjort raskere, særlig med en mer lazy evaluation. Slik den er nå hadde vi det bare som regel at fitnessen alltid skal være oppdatert, så den evalueres hver gang vekt eller alder endres på dyra. Vi kunne også prøvd mer rundt med jit.

**Side 5+6+7 Testen av Carnivore\_feeding [J]**

Her vil vi på en bug som vi oppdaget under testing av carnivorenes spise-metode. Kan dere se forskjell på disse to bildene av koden?

Feilen var en >, der det skulle vært den >=.

Her er testen. Den er rigget slik at carnivorene skal lykkes i alle jakten, men den ene herbivoren ville ikke bli spist. Vi prøvde med å legge til flere både carnivorer og herbivorer, men den ville ikke dø. Men så fant vi feilen nå vi så nærmere på lista med boolske verdier som ble returnert av carnivorens metode.

**Side 9 doc forside [J]**

Her er forsiden av vår dokumentasjon. Vi har laget den med å skrive doc-strings i numpy sin stil, som så er gjort om til html-kode med Sphinx. Her er det en side for hver modul, samt søkefunksjoner og slikt.

**Side 12 eksempel/resultet fra mini-øy [J]**

Som vi så i check\_sim eksemplet får vi oscillasjoner i bestandene etter litt tid. Det skjer også i dette eksemplet, selv med en veldig anneledes øy. Den er mye mindre, samt at jungel parameteren for max matmengde er økt fra 700 til 3000. Den økningen i mat gjør at osilasjonene blir mye skarper/bråere. Med en gang det er få carnivorer skyter herbivorene i vært, før veksten stoppes av en økning i carnivortallet. På denne øya starter vi også med carnivorer, men det gjør ikke noe på lang sikt.