«Bowling»

Prosjekt1

# Oppgave:

Dere skal simulere et bowling-spill. Dere skal programmere deres egen løsning av dette og det finnes ingen fasit. Men det kommer til å være noen krav lenger ned.

Del 1: Gjennomgang, Planleggingsfase og oppstart; Tid: tirsdag 29 SEPTEMBER   
Tegn ned ideer til hvordan ditt program skal se ut. Skriv opp hva slags viktige momenter du ønsker å ha med. Mulige eksempler:

* 1 eller flere spillere
* Poeng
* Runde
* Hastighet
* Kunne skru ballen på noen måte?
* Grafikk eller bare tekst

Forventer her tekst og en sketch av systemet gjerne for hånd og så ta bilde eller lage det i et tegneprogram. Kan starte med koding hvis ideen er godkjent av lærer. Ingen grupper, dere bytter plasser før prosjektet er over. Men samarbeid gjerne underveis med tankegang, men ikke lever den samme løsningen.

## Del 2: Utførelse:

Jobb i Visual Studio Code med koden deres. Eventuell grafikk kan løses i PyGame. Fremføring 1 november i timen.

# Krav og kriterier:

* Alle regler i bowling skal være med
  + 10 runder (frames)
  + **Runde 1-9** har maks to kast og minimum 1
    - Ved strike på 1. kast per runde så ender runden etter kun 1 kast
  + **Runde 10** har maks 3 kast og minimum 2.
    - Ved strike på første kast så har man alltid 2 til (3 kast til sammen)
    - Ved spare på 2. kast så får man et kast til (3 kast)
    - Ved 3 strike i 10ende runde så blir det fortsatt bare 3 kast.
* Alle kast kan enten beregnes på en type input som dere bestemmer eller den kan være random basert på noe lignende **random(0, antall\_pinner\_igjenn).**
* Vi skal kunne se hva som foregår mellom vært kast. Minimum er en type grafikk som viser score per runde som vanlig bowling. Men kan løses med bare tekst.  
  [0][6] [2][7] [X] [9][/] [3][1] etc  
   6 15 35 48 52  
  **Obs! Det er riktig poeng over i eksempelet**
* Minimum poeng er 0 (ved 20 bom), maks poeng ved 12 strike på rad er 300.
* Alle skal bruke disse best practice reglene:
  + snake\_case eller camelCase
  + liten bokstav på variabler, funksjoner og klasser
  + dere skal **ikke bruke global** foran variabelnavn
  + en funksjon skal helst bare gjøre en ting (denne må vi diskutere mer i klassen, men prøv å følge denne regelen der det gir mening. Eksempel funksjonen send\_ball(hastighet) skal ikke printe ut poengsum, men skal da heller kalle på funksjonen print\_poengsum() fra send\_ball(hastighet) funksjonen.
* Relativt korte navn, men navnene skal gi mening
  + Eksempel på gode navn:
    - oppdater\_poeng()
    - neste\_runde()
    - nytt\_spill(antall)
  + Eksempel på dårlige navn:
    - funksjon1(f)
    - funksjon\_for\_å\_oppdatere\_poengsum(ny\_poengsum, runde)
    - e(f)
* Koden skal kommenteres kort der det trengs på det meste. Mer om dette i løpet av prosjekttiden.
* Spillet/simuleringen skal kunne lagres underveis

### Fremføring

Vi prøver å holde maks 3 minutter per person. Snakk om det du er mest stolt av og det du har hatt mest problemer med. \*Ingen karakter på fremføringsdelen.

### Mulige ting dere Kan snakke om og vise fra arbeidet deres:

* Ting dere er spesielt stolte av
* Løsninger som ikke alle andre har
  + PyGame
  + Animasjon
* Ting som har vært spesielt vanskelig

## Kompetansemål:

De i blått vil alle være innom, de andre kan dere se om dere får vist noe kompetanse innom selv.

* utforske og vurdere muligheter, utfordringer og konsekvenser ved bruk av informasjonsteknologi i ulike sammenhenger
* drøfte etiske dilemmaer som oppstår som en konsekvens av informasjonsteknologi, både for individ og samfunn
* utforske og vurdere alternative løsninger for design og implementering av et program
* anvende objektorientert modellering til å beskrive et programs struktur
* utvikle objektorienterte programmer med klasser, objekter, metoder og arv
* vurdere og bruke strategier for feilsøking og testing av programkode
* generalisere løsninger ved å utvikle og bruke gjenbrukbar programkode
* vurdere brukervennligheten i egne og andres programmer og foreslå forbedringer
* velge og bruke relevante systemutviklingsmetoder og -verktøy for samarbeid med andre
* gjøre rede for standarder for lagring, utveksling og sikring av ulike typer data
* bruke programmering til å innhente, analysere og presentere informasjon fra reelle datasett

# Vurdering / innlevering

Kun fremføring uten karakter. Men blir tilbakemelding fra lærer i timen 😊

**Regler for bowling kort**

* 10 runder (frames)
* 2 kast per runde (noen unntak)
* Du får poeng per kjegle du velter i en runde
  + Strike og Spare modifiserer regelen over
* **Strike** er når du får ned alle kjeglene på første kast i en runde (unntak fra regelen i runde 10). Merkes med X og runden er over (unntak runde 10). [x]
* **Spare** er når du får ned alle kjeglene i kast nummer 2 i en runde og merkes med en / istedenfor et poeng [2][/] betyr 2 kjegler på første kast og resten(8) på andre kast.

Modifisering av poeng:

* Strike
  + Strike i en runde gir 10 poeng pluss neste to kast.
    - Strike med Spare etterpå gir 20 i den runden du fikk strike.  
      [X] [9][\]
    - Strike, strike, 5 gir 25 i første strike. Vi har ikke nok data ennå til strike nummer 2.  
      [X] [X] [5][?]
* Spare
  + Spare gir 10 poeng pluss kastet etter.
    - Spare, 6, 3 gir 16 i den runden spare ble kastet.  
      [3][\] [6][3]
    - Spare, strike gir 20 der spare ble kastet.  
      [0][\] [X]
* Runde 10, Spillet legger til et ekstra kast ved behov. 3 kast holder i alle scenarioer
  + [X] gir 2 kast til uansett.
    - [X][4][\] = 20
    - [X][X][3] = 23
    - [X][X][X] = 30 (maks)
  + [3][\] spare gir et kast ekstra
    - [3][\][0] = 10
    - [3][\][3] = 13
    - [3][\][X] = 20

Øv her til du skjønner hvordan poeng fungerer:

<https://www.bowlinggenius.com/>

Bruk denne også til å teste om poengene dine er riktig underveis i utviklingen.

Poengsystem på engelsk:

<https://www.youtube.com/watch?v=7rb_Bhvh8k4>