Oblig 4 – Programmering 1

TEORI:

- Teorioppgave 1: *Forklar hva en exception er, hvordan vi kan håndtere dem og i hvilke tilfeller slik håndtering kan være relevant.*

En exception er det man kan kalle for en runtime error, altså noe som kan gjøre at programmet krasjer. Relatert til denne obligen kan en exception være f.eks. når blackjack-spilleren blir spurt om input, velg 1 for play (ny runde) og 2 for quit. Hvis man da taster in feil, f.eks 3 eller bokstaver så vil programmet krasje, runtime error, grunnet denne exception. For å kunne håndtere slike «feilkilder» hvor det er fort gjort å krasje programmet ved feil input, kan vi bruke det som heter «Try – Except». Vi setter den koden vi ønsker innenfor en try/except blokk som håndterer, eller fanger opp, feilen vi eventuelt kan gjøre. (I andre programmeringsspråk heter det ofte try/cath, noe jeg personlig syns gir mer mening). Det finnes mange typer exceptions vi kan fange opp. Mest relevant for denne obligen er kanskje ValueError, noe som skal fange opp ikke gyldig nummer(input). F.eks. har du kun valg mellom 1 og 2, men taster 4, uten en try/except vil programmet krasje – du får ValueError hvis ikke det er definert hva som skal fanges opp. Eller, uten definisjon vil den fange opp alle feil, men dette er ikke best practice.

- Teorioppgave 2: *Forklar med egne ord hva en klasse er. Gi gjerne et eksempel eller to.*

En klasse kan vi bruke hvis vi ønsker å definere en datatype som vi ikke kan definere ved hjelp av «vanlige» datatyper, som booleans, strings, lister og nummer, osv. Relatert til denne obligen kan vi lage en klasse for f.eks. en spiller og en dealer. Vi lager altså vår egen datatype. Vi kan så definere attributter til disse klassene. Attributter er variabler som er knyttet til klassen vi lager. Som f.eks klassen Player som har attributter som score og hand. Klasser kan ses på som plantegninger (liker bedre ordet Blueprints), men vi kan si plantegninger for denne gangen. Basert på fastsatte plantegninger kan du så lage objekter av disse tegningene.

- Teorioppgave 3: *Forklar med egne ord hva et objekt er, og hva dets relasjon til klasser er.*

Et objekt er en instans av en klasse, altså en utgave av klassen. Basert på plantegningen (klassen) har vi nå laget et objekt av denne plantegningen. I mitt tilfelle har jeg laget klassen player og tilhørende objekt player1. Denne player1 har attributter som score og hand, samt chips. Relasjonen mellom de to kan gjerne forklares med en metafor (syns det hjelper på å få slike abstrakte ting litt mer forståelig). Klassen er en kakeform med dens unike fasong. Det er et verktøy som definerer formen og designen på en kake. Selve kaken er så objektet som vi har brukt kakeformen til å lage. Du kan lage masse forskjellige kaker med denne formen, hver og en kan være helt unik, en med topping, en uten, en med fyll, osv. Dette gjenspeiler f.eks. klassen Player og objektet player1.

- Programmeringsoppgave 2: *Dokumentasjon:*

Jeg valgte å gå for fremgangsmåte 1 i denne oppgaven, altså forsøke å løse denne oppgaven på egen hånd så langt som mulig.

Jeg fikk mye hjelp av eksempelutskriften i oppgaveteksten følte jeg. Så mye på den i starten og begynte å skrive kodeelementer utfra den eksempelutskriften. Men aller først viste jeg at jeg måtte lage klasser. Så jeg laget klassene Dealer og Player og la til attributter til disse. Attributtene ble litt endret på i tidlig fase for å få riktige attributter på hver klasse. Jeg tenkte også tidlig at jeg skulle lage en slags blackjack-klasse (akkurat hva jeg tenkte husker jeg ikke..) men dette endret seg til å ble i stedet til en klasse for kort (Card-klassen). Utfra denne klassen skulle jeg da lage objekter der hvert eneste kort er et objekt av klassen Card.

For å lage selve kortstokken laget jeg to lister som holdt på rank og suit, deretter ble det en for-løkke for å lage en kortstokk (deretter ble dette til 6 kortstokker (Casino-Standard)).

Det første stedet hvor jeg møtte på en utfordring var etter at kortstokk og klasser var laget og første kort i programmet skulle deles ut. Da utskriften kom ble hvert card-objekt (kortene) printet ut på en veldig «ikke lesevennlig måte». Noe jeg leste meg til var den måten objekt ble printet ut på dersom man ikke definerer en metode i klassen som er enten __repr__ eller __str__. Så, akkurat her måtte jeg spørre chatGPT om hva i all verden denne utskriften var.

(til info: måten jeg bruker chatGPT på for hjelp er å alltid be om å få «en dytt i riktig retning» når jeg spør om ting. På denne måten får jeg ikke svar direkte, men må tenke selv basert på de ledetrådene eller hints jeg får til svar. Dette utfordrer meg til å løse ting mest mulig på egen hånd og blir mye mer likt det å bruke en studentassistent (som heller ikke bare gir deg svaret). Denne måten er for meg veldig lærerik.)

Så, til slutt skjønte jeg (med hjelp) at jeg måtte definere en annen måte for objektene å printes ut på, og dette ble __repr_metoden (som returnerer en string som representerer objektet som skal printes). Etter att denne metoden var definert fungerte dette fint.

Når det kom til check for blackjack gikk det litt ekstra tankekraft. Var usikker på akkurat hvilken metode som kunne bli best for å utføre sjekken, og tenkte først at jeg skulle prøve å håndtere ess som både 1 og 11, men det ble til slutt en relativt enkel sjekk med å iterere gjennom spillerhånd og legge til enten «face_card» eller «ace» i en counter-variable for å så legge til i denne variabelen dersom spilleren hadde face card og/eller ess på hånda. Sjekken/programmet behandler ess kun som 11.

Til sist ble det litt mange sjekker for å avklare vinner/split(likt resultat) osv. Fikk da god hjelp med parprogrammering av samboer som er informatiker. Det er gull verdt å ha denne hjelpen, ikke minst ha 2 ekstra øyne som kan se på koden.

Det som kunne vært bedre:

- Flere blokker med kode ekstrahert til funksjoner/moduler (for ryddigere kode).
- Ess behandlet som både 1 og 11.
- Fikset detaljer som prompts for å spille igjen(hvor det nå mangler)

- Fikset slik at kortstokkene re-initialiserte seg (fikk 312 nye kort) for hver runde. Prøvd på dette ved å lage funksjon av selve genereringen av kortstokkene, men ikke fått dette til i programmet.

Oppsummert er jeg likevel fornøyd. Jeg kunne gjerne brukt mer tid på å fikse det som kan fikses, men tar rådet til Nils-Christian om å ikke bruke mer tid enn nødvendig på denne obligen siden neste er rett rundt hjørnet. Det har vært en veldig lærerik prosess.