Teorioppgaver Oblig 4 – Titas Palijanskas

Teorippgave 1.

En exception er blokk av kode som kjøres når kode som er ustabil blir kjørt, men det er ikke alltid slikt at den koden kan kjøres trygt, derfor er vi nødt til å ha kodeblokker som try..except. Dette nøkkelordet vil kjøre en slags test av koden som trenger et ekstra lag med sikkerhet, og hvis den feiler vil except fange dette. Slike håndteringer er spesielt viktige I tilfeller der vi jobber med mye kode, og vil ha en tilpasset error for det slik at vi vet nøyaktig hvor problemet ligger.

Teorioppgave 2.

En klasse er en form for bygging av objekter med egendefinerte variabler og funksjoner knyttet til klassen. Altså en hund kan være en klasse, variablene kan være alder, rase, navnet til eieren osv. Funksjonene som kan utføres kan være at hunden spiser, leker, samhandler med andre hunder o.l.

Teorioppgave 3.

Et objekt er en del av en klasse, altså det er en forekomst av en klasse der den har blitt tildelt verdier og har muligheten til å utføre eksisterende funksjoner. Det kan finnes teoretisk sett uendelig mange objekter I en klasse, og objektet kan knyttes til flere klasser I form av arv. Et hund-objekt kan være en del av hund-klassen, men også en kjæledyr-klasse, og til slutt et dyr-klasse.

Dokumentasjon – Oppgave 2:

Jeg begynte med å lage en dictionary av alle kortene slik at jeg kan sette verdiene på en ordentlig måte, men jeg merket at dette bruker man mye tid på, og er litt upraktisk. Derfor skrev jeg en for-loop som tar listen med korttypene og lager hvert nummer for hver type, samt tildele verdiene. Etter dette begynte jeg på den initiale while-loopen som tar inn brukerens nåværende chips, og tar en beslutning basert på det om brukeren har tilstrekkelige chips etter forespørsel.

Neste delen var det å konstruere en egen metode for å mikse kortene. Jeg søkte litt opp på nett og avgjorde at det var best å liste nøkleverdiene, altså kortene i en array slik at de er synlige for brukeren. Disse ville da legges i hendene, og deretter fjerner fra den originale listen slik at man ikke får to eller flere av det samme kortet. Den enkleste delen av denne ble det å regne håndverdi. Jeg slet litt med å komme frem til en løsning på hvordan jeg skulle få til at «ess» ble en 1-er eller 11, men jeg lagde en teller for dette som gjorde at man fikk mindre poeng dersom det var mer enn en ess på hånden.

I hoveddelen av programmet hadde jeg flere hindringer, men de ble løst etter hvert. Den viktigste var å komme seg ut av funksjonen etter at en av «flaggene» (checkbust, checkwin eller checkblackjack) ble merket, og det som stoppet progressen var det at den kom tilbake til session funksjonen selv om en av flaggene ble hevet. Løsningen var det å returnere en boolean påstand etter hver funksjon, og komme seg trygt ut av funksjonen etter at den ble utført. En annen utfordring var det at jeg forsøkte å resette hendene ved bruk av (= []) etter at spillet ble ferdig, men for en eller annen grunn gjorde det at hånden kom aldri opp. Jeg løste dette ved å bruke .clear() funksjonen for listene istedenfor.

Til slutt ryddet jeg litt på koden, la til kommentarer, og sikret at alle utfall var trygge og forutsigbare.