IT2-prøve, fredag 14. mai 2024

Generell informasjon

- Vurderingskriterier ligg sist i dette dokumentet
- Hjelpemiddel: Alle, med unntak av kommunikasjon – noko som inkluderer AI/KI. Sjå vurderingskriterier for meir informasjon om kjeldereferansar og kommentarar.
- Zip/komprimer alt innhaldet i arbeidsmappa når du leverer inn arbeidet på slutten av dagen. Navngje zip-fila «Ditt namn.zip»
- Oppgåvesettet består av 3 (tre) stk. oppgåver, der alle skal løysast. Den tredje oppgåva har mindre omfang enn dei to første.
- Det blir ein fagsamtale i etterkant av heildagsprøven.

Oppgåve 1: Databehandling (30%)

Last ned datasettet i vedlegget til oppgåva. Dette er ei CSV-fil: sikkerheit.csv

Ta utgangspunkt i datasettet som inneheld data om ulike tilfeller av registrerte sikkerhetsbrudd i ei bedrift, og løys deloppgåver A-D.

NB: Du skal ikkje bruke Pandas for å løyse denne oppgåva, då gjer du i så tilfelle det **i tillegg** til «vanilla»-Python. Spør om uklart.

Del A

Skriv ein kort forklaring på kva du meiner kan vere problematiske data i eit slikt datasett, og kva fylgjer det kan få.

Gå tilbake til denne deloppgåva og skriv om det du eventuelt oppdagar av problem seinare i oppgåveteksten, og korleis du har handtert det.

Ta hensyn til eventuelle problematiske data når du løyser dei andre deloppgåvene.

Del B

Finn ut kva som er dei mest utsatte protokollane («Protocol») og trafikktypane («Traffic Type»), altså der det er mest rapporterte hendelsar.

Skriv dette ut som tekst, og lag ein passande visualisering for å vise det i tillegg.

Del C

Dei ansvarlege for sikkerheita lurar på om dei skal leggje inn begrensingar på pakkelengder («Packet Length») i nettverksutstyr for å betre sikre seg. Ta utgangspunkt i datasettet og vis ulike intervall (som du meiner er hensiktsmessige) for å indikere kva pakkelengder som er mest nytta. Visualiser funna dine.

Del D

Finn ut kva som er dei 5 mest utsatte operativsystema («Device information»). Her er det rom for å gjere nokre eigne val av korleis du tolkar innhaldet.

Lag ein visualisering av desse data.

Oppgåve 2: OOP, bibliotek (50%)

Du skal lage ein applikasjon som implementerer konsepta me har lært om innan OOP. Du står sjølv ansvarleg for å vise breidden i din kompetanse når du utviklar ei løysing basert på oppgåveteksten under.

Du skal lage ein applikasjon som lar deg handtere ei løysing for eit bibliotek. Biblioteket har bøker, både fysiske og digitale, samt at dei har skriftlege versjonar og lydbøker. Det må vere mogleg å kategorisere desse basert på sjanger/genre, og andre faktorar som du bestemmer.

Minstekrav til informasjon om ei bok (legg til meir etter behov):

- Tittel
- Forfattar, eller forfattarar
- ISBN
- Utgivelsesår
- ...

Det skal vere mogleg å leige bøker, og levere desse inn etter at ein har leigd. I forbindelse med dette er det viktig å ha kontroll på kor mange bøker ein har inne av ei gitt bok til ei kvar tid.

Det skal vere moglegheit for å handtere varsling når utleigeperiode nærmar seg, og ein strategi for å handtere overskridelse av leigeperiode.

Det skal vere mogleg å ha lister over dei mest populære bøkene (typisk basert på kor mange som har lånt).

Biblioteket ynskjer å nytte ei CSV- eller JSON-fil for å lagre data mellom kvar gong programmet køyrer. Programmet må med andre ord handtere å skrive til og lese frå denne fila.

(Om du ynskjer å nytte ein annan form for lagring, som database, står du fritt til det – men det er ikkje nødvendig for høg måloppnåing.)

Del A

Lag ein UML-modell for applikasjonen.
Beskrivelsar/kommentarar legg du i koden, eller annan egna stad.

Del B

Utvikle applikasjonen basert på krava frå oppdragsgjevar, samt eigne vurderingar du gjer.
Begrunn eigne vurderingar i koden (i form av kommentarar), eller annan egna stad.

Del C

Skriv testar for utvalte deler av applikasjonen. Bruk unittest, eller liknande.

Oppgåve 3: Spel i Pygame (20%)

Ta utgangspunkt i finding_teacher_start.py. Fila har ein del kommentarar og hint til kvar du skal leggje til koden du blir bedt om under.

Kort fortalt er konseptet at spelaren skal finne og trykke på eit skjult objekt på skjermen. Objektet flyttar gradvis rundt. I og med at du ikkje ser objektet skal du få hint om kvar det er i form av at det blir spelt av ein høgare lyd jo nærmare spelaren sin muspeikar er.

Dersom du finn kode som du meiner er plassert feil/ulogisk stad, så kan du gjerne flytte på dette.

Legg til fylgjande funksjonalitet:

- Legg til kode for debugging, meir spesifikt at du kan sjå objektet medan du testar koden din. Det bør vere slik at du kan kommentere ut og inn denne linja etter behov.
- Legg til at lydfila «tone.mp3» startar å spele av når spelet startar.
- Juster presisjonen som er nødvendig for å treffe det skjulte objektet, då det oppleves for enkelt for dei som har testa spelet så langt.
- Lat volumet for lydfila bli høgare når muspeikaren/spelaren er nærme det skjulte objektet.
- Begrens tida spelet køyrer til lengden av lydfila – der du til dømes kan nytte time-biblioteket for å løyse dette. Dersom tida går ut før spelaren finn objektet så blir det skrive ut «Du er ein skam.»
- Dersom spelaren finn objektet så avsluttast spelet, og det blir skrive ut «Gratulerar! Du fant det skjulte objektet på X,X sekunder.»

Gjer gjerne fleire forbetringar dersom du får tid.

Vurderingskriterier, høg måloppnåing:

- God struktur (lett å finne fram i koden, kode som høyrer saman ligg typisk samla)
- God navngjeving (navna gjer meining, fylgjer standard, gjennomgåande)
- Kommentarar (typisk der meir avanserte konsept blir brukt, og spesielt når du finn kode andre stader (NB: hugs kjelderef.)).
- Kjeldehenvisningar der kode blir henta frå andre kjelder.
- Implementerer generelt funksjonaliteten frå krava i oppgåveteksten.
- Bruker lister og ordbøker (dictionaries) for å strukturere og samle data.
- Bruker funksjonar for å samle kode som høyrer saman.
- Bruker løkker for å iterere gjennom data og utføre eit arbeid.
- -- (databehandling)
- Identifiserer feil og potensielle problem i datasettet, og kan både beskrive og løyse problema.
- Kan iterere gjennom datasettet og velgje ut aktuelle deler som skal vurderast og behandlast, før kalkulasjonar blir gjort for å løyse oppgåvene.
- Kan lage passande visualiseringar av utvalte deler av datasettet, som er tydelege på kva dei viser fram = mellom anna lettlest, luft ml. element, korrekte data.
- Kan nytte eksterne bibliotek for å teikne koordinater til eit kart.
- -- (OOP)
- Kan planleggje og dokumentere eit prosjekt som nyttar OOP, med klassar og arv.
- Kan implementere eit prosjekt som nyttar OOP, med klassar og arv. Strukturen og oversikten i koden er god, med til dømes eigenskapar/attributt og metodar som er plassert på logiske stader i koden.
- Kan lese frå og skrive til ei CSV- eller JSON-fil.
- Kan skrive testar (unittest/enhetstest) for utvalte deler av programmet.
- -- (Pygame)
- Kan skaffe seg oversikt over eksisterande kode, og leggje til funksjonalitet basert på kravspesifikasjon (oppgåvetekst).
-

Lykke til!