IT2-prøve, fredag 19. april 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 (40%)

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

Ta utgangspunkt i datasettet som inneheld data om Airbnb-lokasjonar (utleigeobjekt), og løyse deloppgåver A-E.

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 billegaste og dyraste «price» for utleigeobjekta, i tillegg til gjennomsnittsprisen. Gjer eventuelle vurderingar av resultata du finn, og eventuelle korreksjonar basert på desse.

Del C

Finn ut kva for «host_name» som har flest utleigeobjekt ute på markedet, og ranger dei 5 stk. største utleigarane.

Lag ein passande visualisering av desse data.

Del D

Finn ut kva for «neighbourhood» som har flest utleigeobjekt på markedet, og ranger dei 5 stk. øvste stadane.

Lag ein visualisering av desse data.

Del E

Legg dei 5 stadene med flest reviews på eit kart, jfr. kartoppgåva frå boka – og eksempelet me har sett på i timane. Spør om uklart.

Oppgåve 2: OOP, samlemani (50%)

Du skal lage ein applikasjon som nyttar OOP.

Ein samlar/hoarder treng hjelp til å lage ein app for å halde styr på samlinga si med all mogleg skjit.

Så langt har samlaren funne ein del eigenskapar som han sjølv ynskjer å ha lagra. I tillegg ynskjer han innspel og eventuelle endringar/korreksjonar frå utviklaren (deg).

Det han samlar på er alt frå fysiske spel til ulike konsollar og datamaskiner, filmar i ulikt format, bøker og teikneseriar, «collectibles» som figurar, t-skjorter, merchandise, plakatar osv. Du kan begrense utvalet noko om du manglar tid.

Samlaren si liste med attributt/eigenskapar:

- Pris, når kjøpt
- Pris, verdt per no
- Moglegheit til å vise moglegheit for eventuell fortjeneste
- Kategoriar for kva type produkt det er
- Moglegheit til å sjå alt innan ein gitt kategori
- Moglegheit til å sjå alle produkt innan ein franchise (til dømes alt innan spel, film, bøker, teiknseriar og merch som handlar om Halo)
- ...

Han 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 (10%)

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

Legg til fylgjande funksjonalitet:

- Når «falling object» kjem utanfor skjermen: «Falling object» skal fjernast, og det skal fjernast eit liv frå totalen til spelaren (byrjar på 10).
- Når muspeikar/»crosshair» og «falling object» treff kvarandre: Det skal spelast av ei lydfil: 'thump.wav'
- Når muspeikar/»crosshair» og «falling object» treff kvarandre: «Falling object» skal fjernast.
- Endre kor hyppig «falling object» dukkar opp for å gjere spelet meir utfordrande.
- Avslutt spelet dersom det ikkje er fleire liv igjen.

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!