Systemutveckling i Python – Individuell slutuppgift

Studentinformation

Namn: Anton Sätterkvist
Klass: DOE25
Datum: 14 Oktober 2025
GitHub-länk till projektet: https://github.com/AntonSatt/final-project-python-doe25

---

1. Inledning

I den här uppgiften har ska vi göra ett övervakningsprogram i Python som samlar in den lokala datorns systeminformation som CPU, Minne och Lagring. Det ska kunna övervaka och larma om värden överskrider värden som vi bestämmt i larmen.

Syftet är att få en bra förståelse hur Python programmering fungerar och lära sig baskunskaparna vi kommer behöver att automatisera projekt senare i vår DevOps utbildning.

---

2. Planering och design

Min planering kunde varit bättre och ska bli bättre i framtiden. Jag gick rätt in på uppgiften och såg först att jag behövde 5 olika val i en menu vilket jag gick i genom en efter en.

Det fick jag äta upp senare när jag insåg att jag behövde flera av samma larm så fick göra om min Alarm fil lite med listor istället (som tillslut blev en lista av dictionaries).

Hur jag ska göra i framtiden är att gå igenom uppgiften och skriva upp möjliga funktioner jag kan behöva och sedan tänka hur jag ska dela upp de i olika filer. Det gör att man får en bra överblick över projektet.

---

3. Programstruktur

Programmet är uppdelat i flera olika filer.
`main.py` innehåller huvudmenyn för programmet och är startpunkten.
`monitoring.py` har hand om insamlandet av systemdata (CPU, minne, lagring), även övervakningsläge och visning av systemdata.
`alarm.py` hanterar larm, lagringen av larm, visning av larm.
`utils.py` är standard funktioner som återkommer ofta i de andra filerna (t.ex pauser, rensar terminalen.)
`logger_config.py` hanterar loggningen, då jag fick problem när jag la den i main.py med import. Bättre att ha den i en separat fil.

---

4. Viktiga funktioner eller klasser

`monitoring.py` - Monitoringen klassen är en central bit av mitt program för där sparar vi systemdata varje gång vi uppdaterar från psutil (CPU, Minne, Lagrning). Jag valde att göra en klass här för att kunna lagra

värden och använda de flera gånger över i andra funktioner och metoder.

`alarm.py` är även och en central del där vi laddar in larm från tidigare körning av kod och vi sparar larmen i en lista av dictionaries för att enkelt kunna senare spara in ny larm och ta bort larm vi inte vill ha. Först använde jag mig en lista för larm, då blev det 3 olika larm-listor vilket gjorde det lite krångligt när jag skulle ta bort larm för VG-kraven. Så bättre att köra med en lista av dictionaries för larmen.

---

5. Bibliotek och verktyg

Lista vilka bibliotek jag använde och till vad.

- `psutil` – läsa systemdata
- `os` – filhantering
- `json` – spara/läsa larm
- `time` – för att kunna göra pauser i programmet så man hinner läsa i terminalen vad som händer.
- `logging` – loggning för programmet
- `datetime` – Datum för loggning
- `sys` – Systemfunktioner

Projekten har versionshaterats via Git och Github under hela projektets uppbyggnad. Jag har främst kört in direkt i main, men för större ändring t.ex. när jag behövde bygga om stora delar av programmet för att få ut print() i klasser så körde jag med branches för att enklare ha kontrol över projektet.

---

6. Testning och felsökning

Jag har testat mitt program manuellt genom att köra det i terminalen. Om det är något jag inte förstått hur jag ska göra det så brukar jag skriva in lite print() i funktionerna under tiden jag bygger dom.
Jag använder try/except där det behövs.
Skulle vara bra om jag hade mer testning, eller kanske automatiska tester som kör på varje funktion för att se till att de alla klara av olika sorter av inmatning.

---

7. Resultat

Jag gillar hur mina larm sparas i en lista av dictionaries istället för en lista för varje larm, känns mycket mer "clean". Förut behövde jag sätta ihop en temporär lista där jag la in alla larm för att sedan ta bort det larm man ville ta bort.

---

8. Reflektion och lärdomar

- Att klasser är väldigt bra för att strukturera kod och enkelt producera nya objekt snabbt.
- Jag har lärt mig att det är VÄLDIGT smart att planera hur du ska bygga projektet ibörjan även om det tar lite längre tid först.
- Git och GitHub är superbra, väldigt enkelt att synca mellan laptop:en och datorn hemma.
- Python är väldigt lättläst, jag har mest gjort C innan.

---

9. Möjliga förbättringar och vidareutveckling

- Formateringen av programmet i terminalen, färger osv.
- Förbättring av kod och funktioner på ett smartare sätt.
- Threading för att ha att koden uppdateras i bakgrunden (även om det inte riktigt behövs, men skulle vara coolt och man kan göra lite mer coola saker med det.)
- Mail-funktion
- Tester av min kod, men pytest eller liknande.

---

10. Sammanfattning

Projektet visar hur man enkelt kan använda sig att Python för att skapa viktiga övervakningsprogram för att hålla koll på sin hårdvara.

Jag har lärt mig om filhantering, Python-syntax, versionshantering, klasser och funktioner i riktiga projekt. Kul kurs!
