Osa V: Arendus ja töövood — täisspikker

Tase: **Edasijõudnu**

Git, GitHub ja töövoog

Peatüki täisspikker

Tase: **Edasijõudnu**

Eesmärk: Git hoiab muutuste ajalugu; erista tööpuud, stage'i ja commit'i ning vaata diff enne üle.

Põhikujud

- `git status` — vaata seis
- `git diff` — muutused tööpuus
- `git diff --cached` — muutused stage'is
- `git add fail.txt` — pane stage'i
- `git commit -m '...'` — tee commit
- `git switch -c parandus` — loo haru
- `git pull --ff-only` — uuenda puhtalt
- `git push -u origin parandus` — saada haru
- `git restore --staged fail.txt` — võta stage'ist

Olulisemad lipud, märgid ja kiirnupud

- `HEAD` — praegune tipp
- `main` — põhirida
- `origin` — kaugrepo
- `stage` — järgmine commit

Pane tähele: Uus jälgimata fail ei ilmu tavalisse git diff vaatesse; pärast git add kontrolli esimese commit'i sisu käsuga git diff –cached.

Edasi: Järgmine loomulik samm: Pythoni venv ja eraldatud keskkonnad.

Osa PDF: ./osa-v-arendus-ja-toovood-spikker.pdf

Pythoni venv ja eraldatud keskkonnad

Peatüki täisspikker

Tase: **Edasijõudnu**

Eesmärk: venv hoiab ühe projekti Pythoni paketid eraldi; kõige olulisem on, et terminal, pip ja IDE näeksid sama keskkonda.

Põhikujud

- `python3 -m venv .venv` — loo keskkond
- `source .venv/bin/activate` — aktiveeri shellis
- `python -m pip install -U pip` — uuenda pip
- `python -m pip install requests` — paigalda pakett
- `python -m pip list` — vaata pakke
- `command -v python` — kontrolli tõlgendit
- `deactivate` — välju keskkonnast

Olulised märgid

- `(.venv)` — aktiivne keskkond
- `.venv/` — projekti sees
- `python -m pip` — kindlam kui pip
- `Docker` — terve keskkond

Pane tähele: Kui pip install läks valesse kohta, kontrolli kohe command -v python ja command -v pip; enamasti on probleem just vales aktiivses keskkonnas.

Edasi: Järgmine loomulik samm: Dockeri alused.

Osa PDF: ./osa-v-arendus-ja-toovood-spikker.pdf

Dockeri alused

Peatüki täisspikker

Tase: **Edasijõudnu**

Eesmärk: image on käivitusmall; konteiner on sellest tehtud töötav eksemplar; andmed püsivad ainult hostis, volume'is või Git-repos.

Põhikäsud

- `docker --version` — kontrolli olemasolu
- `docker run --rm alpine echo tere` — ohutu esimene käivitus
- `docker build -t rakendus .` — ehita image
- `docker run --rm rakendus` — käivita image
- `docker ps` — töötavad konteinerid
- `docker logs -f nimi` — jälgi logi
- `docker exec -it nimi sh` — sisene töötavasse konteinerisse
- `docker compose up -d` — teenused taustale
- `docker compose down` — peata komplekt

Olulisemad lipud, märgid ja kiirnupud

- `image` — käivitusmall
- `konteiner` — töötav eksemplar

- `Dockerfile` — image'i retsept
- `compose.yaml` — mitu teenust
- `volume` — püsiv andmesisu
- `registry` — image'ite hoidla
- `--rm` — kustuta lühikatse konteiner pärast lõppu
- `-p 8000:8000` — seo hosti port konteineri pordiga
- `--mount type=bind,src="$PWD",dst=/app` — jaga praegune kaust konteinerisse
- `-w /app` — määra konteineri töökaust
- `down -v` — peatab ja kustutab ka volume'id

Pane tähele: Kui konteiner ei tööta, vaata enne docker logs; ära käivita tundmatut image'it oma kodukataloogi või saladuste bind mount'iga.

Edasi: Järgmine loomulik samm: IDE-d ja arenduskeskkonnad.

Osa PDF: ./osa-v-arendus-ja-toovood-spikker.pdf

IDE-d ja arenduskeskkonnad

Peatüki täisspikker

Tase: **Edasijõudnu**

Eesmärk: hea arenduskeskkond tähendab, et terminal, Git, interpreter ja vajadusel konteiner osutavad samale projektile, mitte eri maailmadele.

Kontrollid ja valikud

- `python3 --version` — kontrolli Pythonit
- `python3 -m pip --version` — kontrolli pip-i
- `git --version` — kontrolli Git-i
- `docker --version` — kontrolli Dockerit
- `node --version` — kontrolli Node'i
- `npm --version` — kontrolli npm-i

Olulised mõisted

- `interpreter` — tegelik käivitaja
- `sisseehitatud terminal` — sama projektivaade
- `Remote SSH` — tööta kaugmasinas
- `devcontainer` — IDE konteineris

Pane tähele: Kui IDE-s "miski ei tööta", kontrolli kõigepealt, kas IDE kasutab sama Pythonit, Node'i või konteinerit mis sinu terminal.

Edasi: Järgmine loomulik samm: Andmeteaduse eelteadmised käsurea vaates.

Osa PDF: ./osa-v-arendus-ja-toovood-spikker.pdf

Andmeteaduse eelteadmised käsurea vaates

Peatüki täisspikker

Tase: **Edasijõudnu**

Eesmärk: andmeteaduse stardis ei piisa ühest tööriistast; vaja on korraga failivormingute tunnetust, SQL-i mõtteviisi, programmeerimist ja statistilist mõtlemist.

Põhikujud

- `fail -> head/less -> column/jq -> sqlite3 -> python3` — alusta väikese vaatusega
- `Pythoni venv` — projekti töölaud
- `CSV, JSON ja XML` — andmete kuju
- `SQLite ja Python` — esimene andmebaas
- `Teksti teisendamine` — väiksed filtrid
- `Vood ja tabelid` — koondamine

Põhiteljed

- `programmeerimine` — korduv töö loogikaks
- `SQL` — küsi ja seo andmeid
- `vormingud` — CSV, JSON, XML
- `statistika` — anna tähendus

Pane tähele: Ära mine otse mudeli või notebook'i juurde enne, kui oled vähemalt korra kontrollinud, mis kujul andmed sul tegelikult on.

Edasi: Järgmine loomulik samm: CSV, JSON ja XML käsureal.

Osa PDF: ./osa-v-arendus-ja-toovood-spikker.pdf

CSV, JSON ja XML käsureal

Peatüki täisspikker

Tase: **Edasijõudnu**

Eesmärk: vormingu tüüp ütleb, milline tööriist mõistab andmeid kõige loomulikumalt: tabelit, puud või märgendistruktuuri.

Põhikujud

- `head fail.csv` — vaata CSV algust
- `column -s, -t < fail.csv` — joonda lihtne CSV
- `cut -d, -f1 fail.csv` — võta üks veerg
- `python3 -m json.tool fail.json` — vorminda JSON
- `jq '.students[].name' fail.json` — vali JSON välju
- `xmllint --format fail.xml` — vorminda XML

Olulised mõisted

- `CSV` — lihtne tabel
- `JSON` — objektide puu
- `XML` — märgendipuu
- `jq` — JSON filtriks

Pane tähele: Kui CSV-s on jutumärgid, komad välja sees või reavahetused, ei pruugi cut -d, enam piisata; siis mine pigem Pythonisse või spetsiaalsema parseri juurde.

Edasi: Järgmine loomulik samm: Andmebaasi algus: sqlite ja Python.

Osa PDF: ./osa-v-arendus-ja-toovood-spikker.pdf

Andmebaasi algus: sqlite ja Python

Peatüki täisspikker

Tase: **Edasijõudnu**

Eesmärk: SQLite on sild tekstifailide ja päris andmebaasimõtte vahel: failina lihtne, SQL-i mõttes siiski relatsiooniline andmebaas.

Põhikujud

- `sqlite3 andmed.db` — ava või loo fail
- `sqlite3 andmed.db '.tables'` — vaata tabeleid
- `sqlite3 andmed.db '.schema'` — vaata struktuuri
- `sqlite3 andmed.db 'select * from students limit 5;'` — piilu ridu
- `sqlite3 andmed.db 'select city, count(*) from students group by city;'` — koonda read
- `sqlite3 andmed.db 'select s.name, r.score from results r join students s on s.id = r.student_id;'` — ühenda tabelid
- `python3 naide.py` — kasuta Pythonist

Olulised mõisted

- `primary key` — rea unikaalne id
- `foreign key` — viide teise tabelisse
- `JOIN` — seo tabelid
- `GROUP BY` — koonda read

Pane tähele: Ära tee JOIN-i ainult käsu pärast; enne joonista enda jaoks välja, milline väli viitab millisele tabelile.

Edasi: Järgmine loomulik samm: Kompileerimine ja käivitamine: shell, Python, C, C++, Go, Rust, Java.

Osa PDF: ./osa-v-arendus-ja-toovood-spikker.pdf

Kompileerimine ja käivitamine: shell, Python, C, C++, Go, Rust, Java

Peatüki täisspikker

Tase: **Edasijõudnu**

Eesmärk: mõni fail käivitatakse tõlgendiga otse; mõni fail kompileeritakse kõigepealt teise vormi; mõni tulemus on päris binaar

Põhikäsud

- `go` — Go tööriist
- `cargo` — Rusti tööriist
- `python3` — käivita Python
- `cc` — kompileeri C
- `javac` — kompileeri Java
- `java` — käivita JVM-is

Tüüpilised kujud

- `chmod +x hello.sh` — õigused
- `python3 hello.py` — käivita Python
- `cc hello.c -o hello-c` — kompileeri C
- `go run hello.go` — Go tööriist
- `go build -o hello-go hello.go` — Go tööriist
- `mkdir -p hello-rust` — loo puuduv rada

Pane tähele: Kui käivitatav fail "ei tööta", kontrolli kõigepealt, kas ta üldse ehitati valmis, kus ta asub ja kas pead kasutama kuju ./fail.

Edasi: Järgmine loomulik samm: LaTeX käsurealt.

Osa PDF: ./osa-v-arendus-ja-toovood-spikker.pdf

LaTeX käsurealt

Peatüki täisspikker

Tase: **Edasijõudnu**

Eesmärk: LaTeX-i mõte on lihtne: lähtetekst jääb tekstifailiks, aga buildist sünnib väga täpselt juhitav PDF.

Põhikujud

- `pdflatex tere.tex` — kompileeri PDF
- `xelatex tere.tex` — parem Unicode tugi
- `latexmk -pdf tere.tex` — korda vajalikud käigud
- `latexmk -xelatex tere.tex` — xelatex automaatselt

- `ls tere.*` — vaata abifaile
- `open tere.pdf` — ava tulemus

Olulised failid

- `.tex` — lähtefail
- `.pdf` — tulemus
- `.aux` — ristviidete abi
- `.log` — kompileerimislogi

Pane tähele: Kui ristviited või sisukord ei ilmu kohe õigesti, ei tähenda see tingimata viga; LaTeX vajab sageli rohkem kui üht kompileerimiskäiku.

Edasi: Järgmine loomulik samm: Lisa A: kopeeritavad minitestid.

Osa PDF: ./osa-v-arendus-ja-toovood-spikker.pdf