


# GitHub Copilot Edits opetukseen

10.12.2024

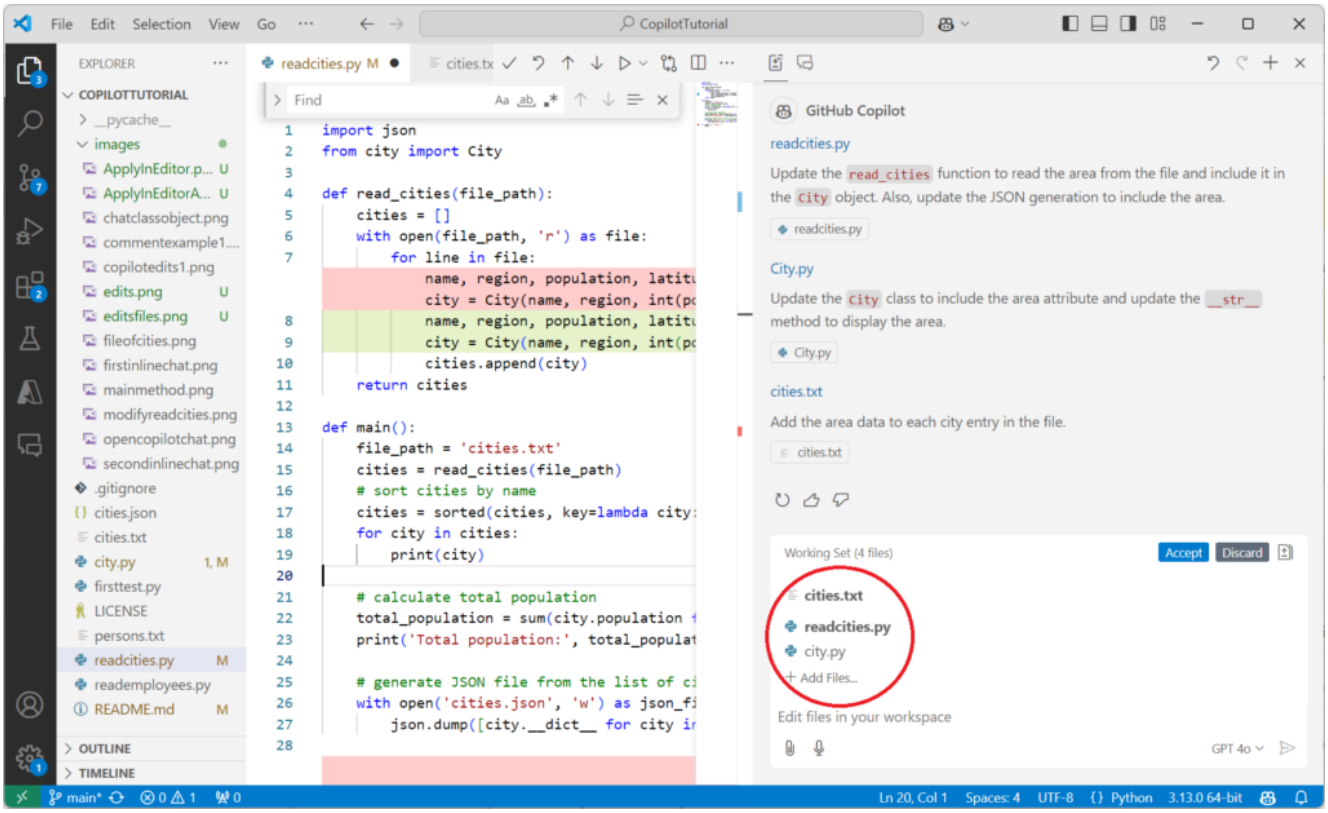
GitHub Copilot -koodiavustaja on tuttu tekoälyyn pohjautuva työkalu useimmille ohjelmistosuunnittelijoille (Dohmke, 2023). Copilotille kuvataan ohjelmointitehtävä luonnollisella kielellä ja se tuottaa ratkaisun tehtävään halutulla ohjelmointikielellä (Microsoft, 2024a). Tekoälyn avulla tuotetut koodiehdotukset auttavat ohjelmistokehittäjää esimerkiksi vaikeasti muistettavien yksityiskohtien ohjelmoinnissa, ja ohjelmistokehittäjä voi keskittyä haastavampiin ohjelmistotehtäviin. Copilot osaa tuottaa koodiehdotuksia myös laajempiin ohjelmistokokonaisuuksiin, jotka koostuvat useista moduuleista.

Copilotille voi antaa syötteitä monilla eri tavoilla. Yhteen tiedostoon kohdistuva muutos tai lisäys voidaan antaa syöteikkunassa (Inline Chat) tai se voidaan kirjoittaa ohjelmakoodin kommentteihin. Copilot ehdottaa myös uusia koodirivejä pyytämättäkin, kun ohjelmoija kirjoittaa ohjelmakoodia. Yleiset koulutehtävät ovat ratkottavissa jopa ilman promptaamista, kun Copilotilla on aavistus, mistä tehtävästä on kyse.

Copilotin Chat-toiminto on pystynyt tekemään myös useampaan tiedostoon kohdistuvia muutosehdotuksia jo pitkään (Microsoft, 2024b). Ohjelmoijan on määriteltävä Chat-ikkunassa itse tiedostot, joita muutospyyntö koskee. Kun pyyntö on annettu, Copilot kertoo tiedostokohtaisesti tarvittavat muutokset Chat-ikkunassa. Ohjelmoijan on siirrettävä ehdotetut muutokset itse kuhunkin kooditiedostoon. Käytännössä tämä on helppoa Chat-ikkunan Apply In Editor ja Insert at Cursor -työkaluilla, jotka siirtävät koodiehdotukset Chatin koodi-ikkunasta editorin puolelle ohjelmakooditiedostoihin. Ohjelmakoodin refaktoroinnin lisäksi Chatin avulla voi tehdä myös kokonaan uusia sovelluksia, jotka koostuvat useammasta tiedostosta.

Marraskuussa 2024 julkaistu GitHub Copilot Edits tuo uusia ominaisuuksia useisiin tiedostoihin kohdistuvien muutosehdotusten käsittelyyn (Microsoft, 2024c). Copilot Edits tekee koodiehdotukset suoraan ohjelmakoodin tiedostoihin, eikä ohjelmoijan tarvitse huolehtia koodin siirtymistä Chat-ikkunasta ohjelmakooditiedostoihin. Alkuperäinen ja ehdotettu uusi koodi näytetään kooditiedostossa (editorissa) eri väreillä korostettuna, mikä

helpottaa koodin tarkastamista (Kuvio 1). Edits pystyy luomaan suoraan myös uusia tiedostoja tarvittaessa. Lisäksi Edits osaa poistaa tarpeettomat metodit, siirtää koodia muihin tiedostoihin, päivittää funktioita käyttämään uusia parametreja ja optimoida koodia suorituskyvyn parantamiseksi.



Kuvio 1. Copilot Edits -näkymä.

Sekä Copilot Chatissä että Copilot Editsissä on keskusteleva käyttöliittymä, joissa voi antaa kehotteita luonnollisella kielellä. Kumpikaan työkalu ei osaa välttämättä päätellä muutettavia tiedostoja itse, vaan ohjelmoijan on kerrottava, mitä tiedostoja muutokset koskevat tai mistä tiedostoista muutokset riippuvat. Suunnittelijan on siis tunnettava ohjelmiston arkkitehtuuri ainakin pääpiirteissään. Tiedostojen lukumäärä on rajoitettu molemmissa työkaluissa. Kontekstiin kuuluvat tiedostot määritellään Chatissä ja Editsissä hieman eri tavoin, mutta periaate on sama. Vaikka Edits on Chatiä kehittyneempi työkalu useisiin tiedostoihin kohdistuvissa muutoksissa, Edits ei kuitenkaan korvaa Chatiä. Chat-näkymässä voi esittää kysymyksiä koodiin liittyen ja kehittää eri tiedostoissa olevia koodilohkoja keskusteluun perustuen, mikä ei taas onnistu Edits-näkymässä.

# GitHub Copilot opetuksessa

Vaikka tekoälypohjaiset koodiavustimet tehostavat ohjelmistosuunnittelijan työtä, ne eivät tee välttämättä ohjelmoinnin opiskelusta sen helpompaa kuin aiemmin. Tekoälyn tehokas hyödyntäminen ohjelmointityössä edellyttää hyvää ohjelmointitaitoa. Lisäksi aloittelevankin koodaajan on tunnettava sovelluksen arkkitehtuuri, ettei hän sotke työkaveriensa töitä. Nyt aloittavan opiskelijan onkin osattava tehdä itse samat ohjelmointitehtävät kuin vaikka viisi vuotta sitten, kun tekoälypohjaiset koodiavustimet eivät olleet vielä laajassa käytössä. Ohjelmoinnin opiskelijan on siis opiskeltava tekoälyn käyttö koodaamisessa kaiken ”vanhan” tiedon lisäksi. Toki joitakin yksityiskohtia voidaan jättää pois opetuksesta ja tekoälyn hyödyntäminen

ohjelmakoodin generoijana ja selittäjänä voi myös tehostaa oppimista.

Tekoälypohjaisten koodiavustimien käyttöä opiskelussa on tutkittu jo jonkin verran. Wermelingerin mukaan perinteistä ohjelmoinnin opetusta tarvitaan edelleen (2023), vaikka ohjelmointi tapahtuu usein tekoälyn avustamana. Denny ym. (2023) selvittivät, miten Copilot selviää tyypillistä ohjelmoinnin perusteiden (Computer Science 1, CS1) tehtävistä pelkän tehtävänannon tai kehotteiden parantamisen avulla. Noin puolet tehtävistä vaativat promptaamisen tarkentamista ja pseudokoodia muistuttava promptaus tuotti parhaat tulokset. Nämä tulokset eivät ole kuitenkaan enää päteviä, sillä Copilot ratkaisee Denny ym. antamat promptauksen tarkennusta vaativat esimerkit suoraan pelkän tehtävänannon perusteella. Nykyään Copilot antaa ratkaisut tunnetuimpien verkkokurssien tehtäviin jopa ilman varsinaista promptaamista. Copilotille riittää, että sillä on aavistus, mistä tehtävästä on kyse. Promptauksen opettamiseen onkin vaikea laatia aikaa kestäviä tehtävänantoja.

Aiemmissa tutkimuksissa on havaittu, että kielimallit eivät osaa yleistää koodia kovin hyvin ja ohjelmakoodin uudelleenkäytössä on ollut puutteita (Harding & Kloster, 2024; Luukkainen, 2024). Tekoälyn yleistyminen ohjelmoinnissa on saattanut vähentää koodin uudelleen käyttöä ja koodin parantamista refaktoroinnin avulla. Tekoälypohjaiset koodiavustimet kehittyvät kuitenkin nopeasti ja esimerkiksi Copilot Edits on tuomassa tähän parannusta. Tosin täytyy muistaa, että Copilotin kontekstin määrittely on edelleen ohjelmoijan vastuulla ja kontekstiin liitettävien tiedostojen lukumäärä on hyvin rajattu.

Copilot Edits tuo kuitenkin uusia mahdollisuuksia ohjelmoinnin opetukseen. Editsin avulla opiskelijalle voidaan tehdä mielekkäitä koodin refaktorointiin liittyviä tehtäviä, jotka opettavat myös ohjelmistoarkkitehtuuria ainakin pienessä mittakaavassa. Opiskelijalle voidaan antaa tarkasteltavaksi esimerkiksi perintähierarkiaa korostava Java-kielinen esimerkki, joka täytyy muuntaa JavaScript-kielelle niin, että ei määritellä luokkia. Pelkkä tekoälyn avulla tehty muunnos ei riitä, vaan opiskelijan on perusteltava vastaus myös aiheesta kirjoitettujen artikkeleiden perusteella.

Tiedämme, että tekoälypohjaiset työkalut tehostavat kokeneen ohjelmistosuunnittelijan työtä, ainakin oikein käytettynä. Sitä emme kuitenkaan tiedä vielä, missä määrin tekoäly auttaa nyt aloittaneita ohjelmoinnin opiskelijoita koodauksen oppimisessa. Vaarana on, että tekoäly tuottaa liian valmiita vastauksia opiskelijoille. Odotan mielenkiinnolla tutkimuksia, joissa verrataan tekoälyn ja perinteisten menetelmien avulla opiskelleiden koodaajien taitoja keskenään. Tekoälyn käyttöä ohjelmoinnissa täytyy tietenkin opettaa, mutta samaan aikaan on varmistettava, että opiskelija pystyy tuottamaan ohjelmakoodia myös itse.

**Petteri Mäkelä**
yliopettaja
automaatio- ja tietotekniikka
SeAMK

# Lähteet

Denny, P., Kumar, V., & Giacaman, N. (2023). Conversing with copilot: Exploring prompt engineering for solving CS1 problems using natural language. *Proceedings of the 54th ACM Technical Symposium on*

*Computer Science Education V. 1*. Presented at the SIGCSE 2023: The 54th ACM Technical Symposium on Computer Science Education, Toronto ON Canada. doi:10.1145/3545945.3569823

Dohmke, T. (2023). The economic impact of the AI-powered developer lifecycle and lessons from GitHub Copilot. *GitHub, Inc*. https://github.blog/2023-06-27-the-economic-impact-of-the-ai-powered-developer-lifecycle-and-lessons-from-github-copilot/

Harding, W. & Kloster, M. (2024). Coding on Copilot – 2023 Data Shows Downward Pressure on Code Quality – 150m lines of analyzed code and projections for 2024. *GitClear.* https://gwern.net/doc/ai/nn/transformer/gpt/codex/2024-harding.pdf

Luukkainen, M. (2024). Kielimallien hyödyntäminen. *Syväsukellus moderniin websovelluskehitykseen: Full Stack open.* Helsingin yliopisto. https://fullstackopen.com/osa1/monimutkaisempi_tila_reactin_debuggaus#kielimallien-hyodyntaminen

Microsoft. (2024a). GitHub Copilot in VS Code. https://code.visualstudio.com/docs/copilot/overview

Microsoft. (2024b). Using Copilot Chat in VS Code. https://code.visualstudio.com/docs/copilot/copilot-chat

Microsoft. (2024c). Copilot Edits. https://code.visualstudio.com/docs/copilot/copilot-edits

Wermelinger, M. (2023). Using GitHub copilot to solve simple programming problems. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. Presented at the SIGCSE 2023: The 54th ACM Technical Symposium on Computer Science Education, Toronto ON Canada. doi:10.1145/3545945.3569830