

# **Tekoälyn vaikutus ohjelmoinnin opetukseen**

Vertailussa Copilot ja ChatGPT



Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus  
kevät, 2023

Laura Mattila

Tietojenkäsittelyn koulutus

Tiivistelmä

Tekijä Laura Mattila

Vuosi 2023

Työn nimi Tekoälyn vaikutus ohjelmoinnin opetukseen  
– Vertailussa Copilot ja ChatGPT

Ohjaaja Lasse Seppänen

---

## TIIVISTELMÄ

Opinnäytetyön tarkoituksena oli selvittää, miten tekoäly tulee vaikuttamaan ohjelmoinnin opetukseen ja kuinka tekoälyä hyödyntävät Copilot ja ChatGPT eroavat toisistaan.

Opinnäytetyön toimeksiantaja oli Hämeen ammattikorkeakoulun tietojenkäsittelyn koulutusohjelma.

Opinnäytetyön teoreettinen osuus koostuu yleisimmistä tekoälyn käsitteistä kuten tekoälyn määritelmästä, historiasta, kehityksestä ja eettisistä ongelmista. Tämän lisäksi tietopohjassa tarkastellaan kahta merkittävää tekoälyohjelmaa, Githubin Copilotia ja ChatGPT:tä, sekä niiden kehitystä, toimintaa, hyötyjä ja haittoja. Opinnäytetyö on toiminnallinen ja tutkimusaineisto kerättiin vertailu- sekä teemahaastattelu menetelmiä hyödyntämällä. Tuloksia havainnollistetaan opinnäytetyössä lainauksien sekä kuvankaappauksien avulla.

Tutkimus osoitti, että tekoälyllä on potentiaalia toimia arvokkaana apuvälineenä ohjelmistokehityksen prosessissa sekä ohjelmoinnin opetuksessa. Kuitenkin tällaisen potentiaalin toteuttamiseksi on tärkeää tunnistaa ja käsitellä tekoälyn eettisiä haasteita sekä varmistaa sen vastuullinen ja turvallinen hyödyntäminen. Näin ollen, tutkimus korostaa tarvetta kehittää ja soveltaa opetusmetodeja, jotka mahdollistavat tekoälyn tehokkaan hyödyntämisen.

Avainsanat Tekoäly, Copilot, ChatGPT

Sivut 57 sivua ja liitteitä 2 sivua

Degree Programme in Business Information Technology      Abstract  
Author      Laura Mattila      Year 2023  
Subject      The Impact of Artificial Intelligence on Programming Education  
                 – A Comparison of Copilot and ChatGPT  
Supervisor      Lasse Seppänen

---

## ABSTRACT

The purpose of the thesis was to investigate how artificial intelligence (AI) will affect programming education and how AI-assisted programs, Copilot and ChatGPT, differ from each other. The thesis was commissioned by Häme University of Applied Sciences' program in information technology.

The theoretical part of the thesis consists of the most common AI concepts, such as AI definition, history, development, and ethical issues. In addition, the knowledge base examines two significant AI programs, Github's Copilot and ChatGPT, as well as their development, operation, benefits, and drawbacks. The thesis is functional, and research data were collected using comparative and thematic interview methods. The results are illustrated in the thesis using quotations and screenshots.

The research demonstrates that AI has the potential to serve as a valuable tool in the software development process and programming education. However, to realize this potential, it is essential to identify and address AI's ethical challenges and ensure its responsible and safe use. Therefore, the research emphasizes the need to develop and apply teaching methods that enable effective use of AI.

Keywords      Artificial intelligence, Copilot, ChatGPT

Pages      57 pages and appendices 2 pages

## Sanasto

AI	Artificial Intelligence, ohjelmistot, laitteistot ja järjestelmiä, jotka kykenevät ihmisten kaltaiseen ajatteluun, oppimiseen sekä rationaaliseen päätöksentekoon
AI pair programmer	Tekoälyä hyödyntävä pariohjelmoija
ANNs	Artificial neural networks, tilastollisia malleja, joiden kehittämisessä on osittain yritetty jäljitellä biologisen neuroverkoston toimintaa
Array	Joukko, tietorakenne, joka koostuu joukosta elementtejä (arvoja tai muuttujia), joista kukin tunnistetaan vähintään yhdellä taulukkoindeksillä.
Block Comment	Lohkokommentti, useamman rivin pituinen ohjelmointikommentti
Boolean	Tietotyyppi kahdella mahdollisella arvolla: true tai false
Codex	Copilotin hyödyntämä kielimalli, joka on kehitetty GPT-3 kielimallista
DL	Deep Learning, syväoppiminen, koneoppimistekniikka, joka opettaa koneita ja ohjelmistoja toimimaan ihmiselle luonnollisella tavalla eli oppimaan esimerkin avulla
GPT-3	Generative Pretrained Transformer 3, Vuonna 2020 julkaistu autoregressiivinen kielimalli, joka käyttää syväoppimista ihmisen kaltaisen tekstin tuottamiseen
GitHub	Koodin isännöintialusta versionhallintaa ja yhteistyötä varten
IDE	Integrated Development Environment, ohjelmointiympäristö, joka tarjoaa kattavan ympäristön ohjelmointikehitykseen
Inline Comment	Rivikommentti, yhden rivin pituinen ohjelmointikommentti
Integer	Tietotyyppi, joka arvo voi olla ainoastaan kokonaisluku
Java	Korkean tason, luokkapohjainen, olio-ohjelmointikieli
JavaScript	Ohjelmointikieli, joka on yksi World Wide Webin ydintekniikoista HTML:n ja CSS:n rinnalla.

LISP	List Progressing, usean lähestymistavan refleksiivinen ohjelmointikieli
ML	Machine Learning, koneoppiminen, jonka avulla ohjelmistosovellukset voivat ennustaa tuloksia tarkemmin ilman, että niitä on erikseen ohjelmoitu tekemään niin
Open Source	Avoinlähdekoodi, vapaasti saatavilla mahdollista muokkausta ja uudelleen jakelua varten
Python	Korkean tason yleiskäyttöinen ohjelmointikieli
The imitation game	Imitaatiopeli, vuonna 1950 esitelty Turingin testi
VScode	Visual Studio Code, koodieditori, joka on optimoitu nykyaikaisten verkko- ja pilvisovellusten rakentamiseen

## Sisälllys

1	Johdanto .....	1
2	Tekoäly.....	3
2.1	Tekoälyn määritelmä .....	3
2.1.1	Tekoälyn historia .....	4
2.1.2	Tekoälyn kehitys.....	7
2.2	Githubin Copilot .....	9
2.2.1	Copilotin kehitys ja toiminta .....	9
2.2.2	Copilotin hyödyt ja haitat.....	10
2.3	ChatGPT.....	12
2.3.1	ChatGPT:n kehitys ja toiminta.....	12
2.3.2	ChatGPT:n hyödyt ja haitat .....	13
2.4	Tekoälyn etiikka ja sen vaikutus opetukseen.....	14
2.4.1	Tekoälyn eettiset ongelmat .....	14
2.4.2	Tekoälyohjelmien vaikutukset korkeakouluissa .....	15
2.5	Tekoälyn hyödyntäminen ohjelmistokehityksessä .....	17
3	Tutkimuksen toteutus ja menetelmät.....	19
4	Copilotin ja ChatGPT:n eroavaisuuksien vertailu .....	21
4.1	Ohjelmointitehtävien suorittaminen Copilotilla .....	22
4.1.1	Ensimmäinen testitapaus: Lucky7.....	25
4.1.2	Toinen testitapaus: Yksikkötestitapausten generointi .....	29
4.1.3	Kolmas testitapaus: Ongelmanratkaisu .....	30
4.2	Ohjelmointitehtävien suorittaminen ChatGPT:llä .....	32
4.2.1	Ensimmäinen testitapaus: Lucky7.....	34
4.2.2	Toinen testitapaus: Yksikkötestitapausten generointi .....	38
4.2.3	Kolmas testitapaus: Ongelmanratkaisu .....	40
4.3	Tulosten analysointi .....	42
5	Copilotin ja ChatGPT: hyödyntäminen ohjelmoinnin opetuksessa.....	45
6	Johtopäätökset ja pohdinta.....	48
6.1	Vertailututkimus .....	49
6.2	Teemahaastattelututkimus.....	50
7	Yhteenveto .....	52

## **Kuvat, ohjelmakoodit ja taulukot (nämä pitää päivittää kuten sisällyskin, poista tarpeettomat)**

Kuva 1 Havainnollistava kuva Turingin testistä (Thompson, 2021).....	5
Kuva 2 Vasemmalla biologinen ja oikealla keinotekoinen neuroni (Addagatla, 2021)....	8
Kuva 3 Havainnollistava kuva Copilotin teknisestä toiminnasta (Prokopets, 2021) .....	10
Kuva 4 Copilotin aktivointi ja deaktivointi.....	23
Kuva 5 Komennon syöttäminen Copilotille Java-kielellä.....	24
Kuva 6 Copilotin generoiman koodin hyväksyminen ja valinta.....	25
Kuva 7 Copilot Lucky7 osa 1 pelin voitto .....	26
Kuva 8 Copilot Lucky7 osa 1 pelin häviö.....	27
Kuva 9 Copilot Lucky7 osa 2 pelin voitto .....	28
Kuva 10 Copilot Lucky7 osa 2 pelin häviö.....	28
Kuva 11 ChatGPT:n käyttöliittymä.....	34
Kuva 12 Lucky7 osa 1: Komennon syöttö ChatGPT:lle .....	35
Kuva 13 Lucky7 osa 2: Komennon syöttö ChatGPT:lle .....	37
Kuva 14 Yksikkötestien generointi: Komennon ja ohjelmakoodin syöttö ChatGPT:lle...39	
Kuva 15 ChatGPT:n generoimat yksikkötestitapaukset .....	40
Kuva 16 Virheellisen ohjelmakoodin syöttö ChatGPT:lle .....	41
Kuva 17 ChatGPT:n löytämät virheet sekä ratkaisuehdotukset.....	42
Ohjelmakoodi 1 Copilotin generoima koodi: Lucky7 osa 1 .....	26
Ohjelmakoodi 2 Copilotin generoima koodi: Lucky7 osa 2 .....	27
Ohjelmakoodi 3 Laskin-ohjelman syöttö Copilotille.....	29
Ohjelmakoodi 4 Copilotin generoimat yksikkötestitapaukset .....	30
Ohjelmakoodi 5 Virheellisen ohjelmakoodin syöttö Copilotille.....	31
Ohjelmakoodi 6 Copilotin löytämät virheet sekä ratkaisuehdotukset .....	32
Ohjelmakoodi 7 Lucky7 osa 1: ChatGPT:n generoima koodi .....	36
Ohjelmakoodi 8 Lucky7 osa 2: ChatGPT:n generoima koodi .....	38

## **Liitteet**

- Liite 1 Aineistonhallintasuunnitelma
- Liite 2 Teemahaastattelujen pääkysymykset



## 1 Johdanto

Teknologia kehittyy hurjaa vauhtia eteenpäin. Samalla markkinoille tulee jatkuvasti uusia julkaisuja sovelluksista sekä ohjelmistoista, jotka hyödyntävät tekoälyä toiminnoissaan. Tekoäly näkyy erittäin vahvasti jokapäiväisessä elämässämme ja sitä hyödynnetään jo esimerkiksi pankeissa, sosiaalisessa mediassa, autoissa sekä lääketieteessä. Aihe on puhuttanut jo useita vuosia, eikä tekoälytrendi tule todennäköisesti laantumaan ajan kuluessa.

Copilot ja ChatGPT ovat tekoälyn ja koneoppimisen uusimpia jättiläisiä ja siksi perehdyinkin tässä opinnäytetyössä niiden ominaisuuksiin, käytettävyyteen sekä ennen kaikkea niiden välisiin eroavaisuuksiin. Opinnäytetyön tarkoituksena on tutkia, kuinka tekoälyä hyödyntävät Copilot ja ChatGPT voivat vaikuttaa ohjelmoinnin opetukseen.

Opinnäytetyön tietoperusta sisältää tekoälyn yleisimmät käsitteet sekä tarkemman kuvauksen Copilotin ja ChatGPT:n ominaisuuksista, joita vertaillaan käytännönsuudessa. Tietoperustassa esitetään myös opettajien ja oppilaiden mielipiteitä tekoälyohjelmien vaikutuksesta opiskeluun ja tuodaan esille tekoälyohjelmien eettisiä ongelmia.

Käytännönsuunnan tavoitteena on testata Copilotin sekä ChatGPT:n eroavaisuuksia erilaisten tehtävien avulla ja tutkia ohjelmien soveltuvuutta ohjelmoinnin opetukseen. Käytännönsuunta sisältää myös haastattelututkimuksen, jossa Hämeen ammattikorkeakoulun ohjelmoinnin opettajilta kerätään mielipiteitä tekoäly ohjelmien soveltamiseen liittyvistä mahdollisuuksista ja riskeistä.

Opinnäytetyön tavoitteena on vertailla ChatGPT:n ja Copilotin ominaisuuksia ohjelmoinnin opetuksen kehittämisen näkökulmasta, kehittää uusia opetusmenetelmiä opetussuunnitelmaan sekä tunnistaa ja minimoida mahdollisia riskejä, joita tekoälyn hyödyntäminen osana ohjelmoinnin opetusta saattaa aiheuttaa. Tutkimus toteutetaan osana Hämeen ammattikorkeakoulun tietojenkäsittelykoulutuksen opetussuunnitelmaa.

Opinnäytetyö vastaa seuraaviin tutkimuskysymyksiin:

1. Miten Copilot ja ChatGPT eroavat toisistaan?
2. Kuinka tekoälyä voisi hyödyntää ohjelmoinnin opetuksessa?
3. Minkälaisia riskejä tai haasteita tekoälyn hyödyntämiseen ohjelmoinnin opetuksessa liittyy?

## 2 Tekoäly

Tekoälyn eli AI:n (Artificial Intelligence) kehitys ja hyödyntäminen arkipäiväisissä toiminnoissa on yleistynyt viime vuosina. Tekoälystä on tullut niin suosittu aihe, että sitä pyritään integroimaan laajasti jo olemassa oleviin järjestelmiin. Useat suuret yritykset ovat sijoittaneet valtavia määriä rahaa tekoälyn kehittämiseen, joka mahdollistaisi kääntämisen, luomisen, rakentamisen, ymmärtämisen ja monien muiden tehtävien koneellistamisen. Koska tekoälyä on vaikea kouluttaa niin, että se ymmärtäisi kaikki sille annetut tehtävät, se voi käytettävyyden kannalta olla joko todella hyvä tai huono työkalu, riippuen kontekstista, jossa sitä hyödynnetään. (Lønvik, 2021)

### 2.1 Tekoälyn määritelmä

Tekoäly on yksi modernin teknologian suurimmista puheenaiheista ja se myös jakaa paljon mielipiteitä. Tekoälylle ei ole yleisesti hyväksyttyä tai täsmällistä määritelmää mutta yleisellä tasolla sillä tarkoitetaan ohjelmistoja, laitteistoja ja järjestelmiä, jotka kykenevät ihmisten kaltaiseen ajatteluun, oppimiseen sekä rationaaliseen päätöksentekoon. Tekoälyä hyödynnetään jo erittäin laajasti monissa arkipäiväisissä sovelluksissa, laitteissa, järjestelmissä sekä palveluissa, joissa tekoälyn tarkoituksena on pyrkiä toimimaan tilanteen tai tehtävän mukaan mahdollisimman optimaalisella tavalla. (Työ- ja elinkeinoministeriön tekoälyohjelman ohjausryhmä, 2017, s. 62)

Vaikka tekoälyä kuvataan usein puhekielessä yhtenä teknologian alana, on sen määritelmä kuitenkin laaja ja moniulotteinen. Tekoälyä hyödynnetään useissa tieteenaloissa ja sen alla on joukko erilaisia teknologioita, sovelluksia, tutkimussuuntia sekä menetelmiä. Tekoälyn kehittyminen on synnyttänyt merkittäviä muutoksia esimerkiksi filosofian, matematiikan, fysiikan sekä tietojenkäsittelytieteen aloilla unohtamatta sen vaikutusta yksilöihin sekä yhteiskuntaan. (Ailisto ym., 2018, s. 6)

Ennen kuin tekoäly voidaan määritellä, on ensin määriteltävä älykkyyden käsite yleisesti. Älykkyyteen viitattaessa puhutaan yleensä mahdollisuudesta oppia, ymmärtää tai soveltaa tietoa yhden tai useamman tavoitteen saavuttamiseksi. Tekoälyä voidaan kuvata koneiden tuottamaksi älyksi ja se rakentuu käsitteistä ja tekniikoista, jotka voivat olla hyvinkin teknisiä

ja monimutkaisia. Ymmärtääkseen täysin tekoälyn määritelmän on tiedostettava sen yhteys matematiikkaan, todennäköisyysteorioihin, koneoppimiseen, psykologiaan, tilastoihin, kielitieteisiin, tietojenkäsittelytieteeseen sekä neurotieteeseen. (Castrounis, 2022)

Käsitteenä tekoäly voidaan jaotella kahteen osaan, vahvaan ja heikkoon tekoölyyn. Tekoälyä hyödynnetään kasvojen- ja puheentunnistusohjelmissa, asiakaspalveluiden chattiboteissa, suoratoistopalveluissa, kohdennetussa mainonnassa sekä lukuisissa muissa sovelluksissa ja ohjelmistoissa. Nämä kaikki ovat esimerkkejä heikosta tekoälystä, joissa tekoäly kykenee vain tiettyjen ennalta määritettyjen ongelmien ratkaisemiseen ja tietyn tasoiseen oppimiseen, sille etukäteen syötetyn datan perusteella. Sillä ei siis voida saavuttaa ihmisen kaltaista tietoisuutta tai ajattelukykyä. Vahvaksi tekoäly määritellään, mikäli se kykenee itsenäiseen ajatteluun ja on ainakin osittain tietoinen itsestään. Ihmisen tavoin vahva tekoäly kykenisi siis itsenäiseen oppimiseen ja olosuhteiden huomioimiseen. Tähän mennessä vahvaa tekoälyä ei olla vielä pystytty kehittämään. (Pietikäinen, 2021, s. 24)

### **2.1.1 Tekoälyn historia**

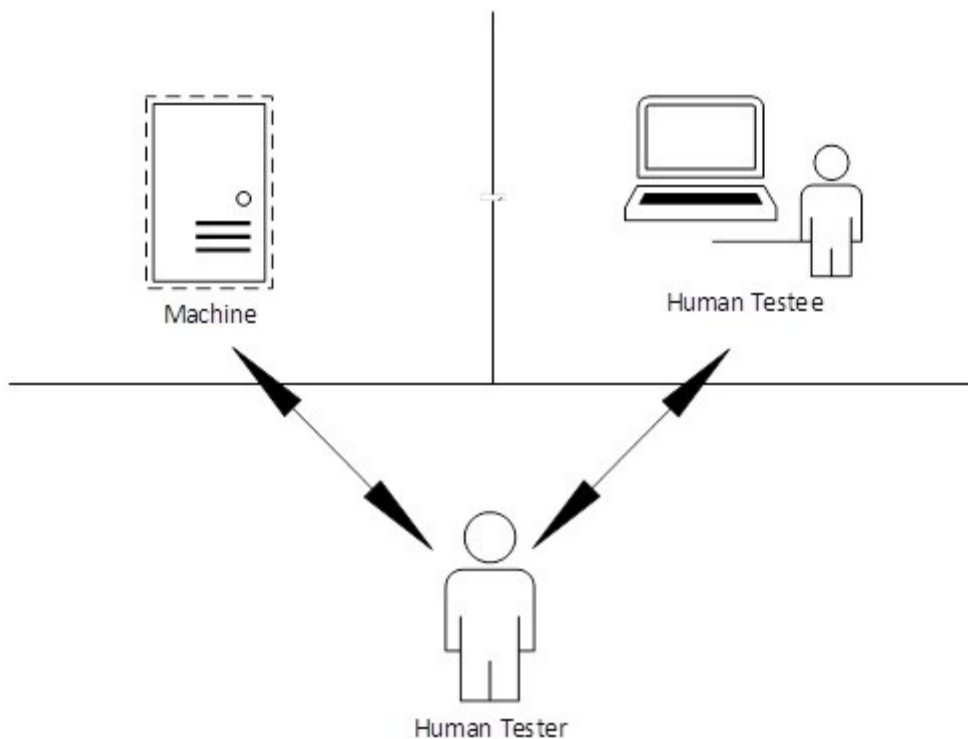
Vaikka tekoäly on vasta viime vuosikymmenenä noussut osaksi yleistä keskustelua ei se kuitenkaan ole käsitteenä uusi, sillä jo antiikin Kreikan kirjallisuudesta on löydetty viitteitä ihmisen halulle kehittää Jumalan kaltainen älyllinen olento. (Pesonen, 2021)

Modernin tekoälyn kehitys katsotaan alkaneen 1940-luvulla, sillä se on suoraan yhteydessä tietotekniikan ja tietokoneiden kehitykseen yleisesti. Vuosilla 1940–1960 on merkittävä vaikutus siihen, kuinka edistynyttä nykypäivän teknologiakehitys on erityisesti tekoälyn kohdalla. Toinen maailmansota ja halu ymmärtää miten voidaan yhdistää kone ja orgaanien olento, kiihdyttivät tekoälykehityksen ajatuksen tasosta konkreettiseksi ideaksi. (Council of Europe, 2023)

Tekoäly niin kuin monet muutkin keksinnöt ovat monen eri henkilön tutkimustyöllä ja keksinnöillä saavutettu lopputulos. Ensimmäisenä varsinaisena tekoölyyn liittyvänä saavutuksena pidetään Warren McCullochin ja Walter Pittsin työtä, jossa he kehittivät keinotekoisien neuroverkkojen mallin ja osoittivat, että tällainen verkko voi olla oppimiskykyinen. (Pietikäinen, 2021)

Brittiläistä matemaatikkoa ja logiikan tutkijaa Alan Turingia pidetään yhtenä tärkeimmistä henkilöistä, joka on ollut mukana luomassa perustaa nykyaikaiselle tekoälylle. Turing tunnetaan parhaiten 1950 vuonna esitellystä Turingin testistä, jota hän kutsui myös nimellä ”imitaatiopeli” (The imitation game). Testi mittaa koneen kykyä käyttäytyä ihmisen kaltaisesti älykkäällä tavalla. Kuvassa 1 on havainnollistettu kuinka testissä ihminen ja tietokone yrittävät vakuuttaa koehenkilön ihmisyydestään. Koehenkilö ei etukäteen tiedä, kumpi vastaajista on ihminen ja mikäli koehenkilö arvaa kysymysten perusteella tietokoneen olevan oikea ihminen on kone tällöin läpäissyt Turingin testin. Yksi ensimmäisiä Turingin testin läpäisseistä oli Joseph Weizenbaumin kehittämä tekoälyohjelma nimeltä ELIZA. (St. George, 2021)

Kuva 1 Havainnollistava kuva Turingin testistä (Thompson, 2021)



Vuonna 1955 amerikkalainen tietojenkäsittelytieteilijä John McCarthy otti termin tekoäly (Artificial Intelligence, AI) käyttöön ensimmäisen kerran ja tästä syystä häntä pidetään niin kutsuttuna ”tekoälyn isänä”. McCarthy aloitteesta järjestettiin vuonna 1956 Dartmounthin kokous, jonka tavoitteena oli selvittää, miten voitaisiin luoda kone, joka kykenee ihmisen kaltaiseen abstraktiin ajatteluun ja ongelmien ratkaisemiseen. McCarthy tunnetaan myös

symbolista laskentaa tukevan LISP-kielen kehittämisestä, josta tuli useiksi vuosiksi tekoälyn hallitseva ohjelmointikieli. (Pietikäinen, 2021, s. 28)

McCarthy lisäksi keskeisissä rooleissa tekoälyn kehitysprosessissa olivat pioneerit Herbert A. Simon sekä Marvin Minsky. Nobel-palkinnonkin voittanut ekonomisti Simon ennusti vuonna 1957, että tietokone tulee voittamaan ihmisen shakissa 10 vuoden kuluttua. Todellisuudessa tähän kului kuitenkin 40 vuotta. Turingin palkinnon voittanut kognitiotieteilijä Minsky taas arvioi vuonna 1967, että tekoälyn ongelmat pystytään pääosin ratkaisemaan yhden sukupolven aikana. (Pietikäinen, 2021, s. 28)

Tekoälykehityksen ongelmien aliarvioinnista seurasi rahoituksen väheneminen sekä radikaali tutkimusten hidastuminen kymmenen vuoden ajaksi. Kehityksen suurimpana esteenä oli laskentatehon puute, jonka seurauksena tietokoneet eivät yksinkertaisesti pystyneet käsittelemään tarpeeksi paljon tietoa tarpeeksi nopeasti. (SITNFlash, 2017)

Esteistä huolimatta 1990–2000 luvuilla useat tekoälyyn liittyvät välietapit oli saavutettu. Vuonna 1997 suurta kohua aiheuttanut shakkiottelu oli ensimmäinen, jossa maailmanmestari Garri Kasparov hävisi IBM:n kehittämälle Deep Blue -nimiselle shakinpeliohjelmalle ja sitä pidettiinkin valtava askeleena kohti toimivaa keinotekoisia älyohjelmaa. Samana vuonna Dragon Systemsin kehittämä puheentunnistusohjelmisto otettiin käyttöön Windowsin käyttöjärjestelmissä ja toiveet siitä, että ei olisi ongelmaa mitä kone ei pystyisi ratkaisemaan olivat suuret. (SITNFlash, 2017)

Vuoteen 2010 tultaessa tekoälyn nousukaudelle oli lähtökotaisesti kaksi merkittävää tekijää. Ensimmäinen oli mahdollisuus päästä käsiksi valtaviin tietomääriin. Ennen algoritmeja esimerkiksi kuvien luokittelussa näytteenotto piti suorittaa itse mutta nykyään yksinkertaisella Google-haulla pääsee käsiksi miljooniin kuviin. Toinen merkittävä tekijä oli tietokoneiden näytönohjainten prosessoreiden kehitys, mikä nopeutti huomattavasti oppimisalgoritmien laskentatehoa. Koska kuvan luokitteluprosessi on erittäin iteratiivinen, se saattoi kestää jopa viikkoja ennen vuotta 2010. (Council of Europe, 2023)

Ohjelmoinnin suhteen ei olla juurikaan viisastuttu tekoälyn kehittämisessä mutta uusien kehittyneiden työkalujen ja tietokoneiden tallennuskapasiteetin ansioista on nykypäivänä

mahdollista hyödyntää entistä monimutkaisempia menetelmiä sekä suuria datamääriä. (Pietikäinen, 2021, s. 7)

### 2.1.2 Tekoälyn kehitys

Algoritmit ovat olennainen osa tekoälyn sekä teknologian kehitystä. Algoritmi tarkoittaa tarkkaa ohjeistusta, jota seuraamalla suoritetaan tehtäviä, prosesseja ja ratkotaan ongelmia. Algoritmit pystyvät muuttamaan ja kehittämään sääntöjä käyttämänsä datan perusteella ja siksi nykypäivänä algoritmeista puhuttaessa viitataan usein oppiviin algoritmeihin. Algoritmeja käytetään monessa tekoälyä hyödyntävässä sosiaalisen median alustassa kuten Facebookissa ja Instagramissa. Algoritmit ratkaisevat missä järjestyksessä ja minkälaista sisältöä nostetaan käyttäjän nähtäväksi esimerkiksi ajan perusteella. (Lampinen, 2019)

Viimeaikoina tekoälykehityksen isossa roolissa on näytellyt koneoppiminen (Machine Learning, ML), joka mahdollistaa tietokoneiden ohjelmoinnin siten, että ne pystyvät niin sanotusti ”oppimaan” esimerkkiedoista sekä aiemmasta kokemuksesta. Ennen koneoppimista tekoäly perustui usein ennalta ohjelmoituihin sääntöihin mutta koska maailmassa ongelmien ja tiedon määrä lisääntyy eksponentiaalisesti on joissain tilanteissa lähes mahdotonta syöttää tekoälylle kaikkea sen tarvitsemaa dataa. (Agrawal ym., 2017)

Koneoppimisen tekniikalla pyritään luomaan tekoälyjärjestelmiä, joilla on kyky toimia järkevästi muuttuvassa ja kompleksisessä ympäristössä. Koska kaikkia tilanteita ei voida etukäteen ohjelmoida tämä vaatii tekoälyltä kykyä oppia, mikäli sovellusalue ei ole tiukasti rajattu. Nykyään tekoäly pystyy oppimaan sille syötetyn datan perusteella ja tekemään päätelmiä myös opetusaineistoon kuulumattomista tapahtumista. (Ailisto ym., 2018)

Nykyisissä tekoälyjärjestelmissä koneoppimisen taso vaihtelee käytetyn menetelmän mukaan. Ohjatussa oppimisessa tekoäly opetetaan ja ohjataan tavoiteltuun suuntaan ja vahvistetussa oppimisessa, oppiminen perustuu vuorovaikutukseen ympäristön kanssa. Valtavaan data-aineistoon ja laskentatehokkuuteen perustuvaa koneoppimista taas kutsutaan syväoppimiseksi. (Ailisto ym., 2018)

Syväoppiminen (Deep Learning, DL) on koneoppimistekniikka, joka opettaa koneita ja ohjelmistoja toimimaan ihmiselle luonnollisella tavalla eli oppimaan esimerkin avulla. Sillä

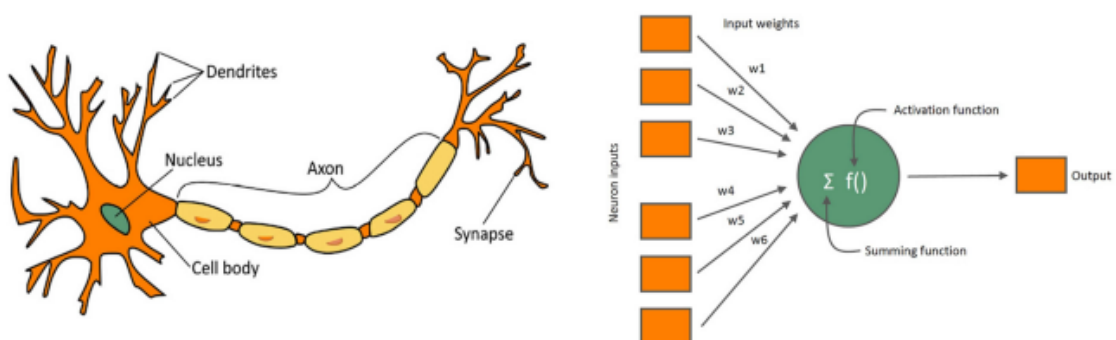
pyritään jäljittelemään tapaa, jolla ihmiset keräävät tietäntyyppistä dataa. Syväoppiminen on tärkeä osa tekoälyn kehitystä sekä datatieteitä ja siitä hyötyvät erityisesti datatieteilijät, joiden tehtävänä on kerätä, analysoida ja tulkita suuria tietomääriä. Syväoppiminen mahdollistaa prosessin suorittamisen huomattavasti nopeammin ja helpommin. Yksinkertaisimmillaan syväoppiminen voidaan ajatella tapana automatisoida ennakoivaa analytiikkaa. Vaikka perinteiset koneoppimisalgoritmit ovat lineaarisia, syväoppimisalgoritmit on pinottu hierarkisesti kasvavan monimutkaisuuden ja ominaisuuksien mukaan. (Burns & Brush, 2021)

Tehokkaimpia tunnettuja tietokoneita ovat ihmisaivot joiden toimintaa havainnollistetaan usein hermosolujen sekä neuroniverkoston avulla. Monien arvioiden mukaan ihmisen aivoissa sijaitsee noin 100 miljardia neuronia, jotka ovat yhteydessä toisiinsa neuroverkoston kautta. (Castrounis, 2022)

Keinotekoiset neuroverkot (Artificial neural networks, ANNs) ovat tilastollisia malleja, joiden kehittämisessä on osittain yritetty jäljitellä biologisen neuroverkoston toimintaa. Neuroverkot ja syväoppivat neuroverkot ovat kehittyneitä algoritmeja, joita sovelletaan tekoälyissä erittäin monimutkaisten ongelmien ratkaisemiseen. (Castrounis, 2022)

Kuvassa 2 on vertailu ihmisen- ja keinotekoisien neuronin eroavaisuuksista. Toisin kuin biologinen neuroverkko, jossa mikä tahansa neuroni voi muodostaa yhteyden mihin tahansa toiseen neuroniin, keinotekoisilla neuroverkostoilla on erilliset kerrokset, yhteydet ja tiedon leviämissuunnat. (Copeland, 2016)

Kuva 2 Vasemmalla biologinen ja oikealla keinotekoinen neuroni (Addagatla, 2021)





## 2.2 Githubin Copilot

Vuonna 2021 Github julkaisi suuria toiveita herättävän Copilotin, jota kutsutaan myös nimellä ”tekoälypariohjelmoija” (AI pair programmer). Microsoftin ja OpenAI:n yhdessä kehittämä syväoppimista hyödyntävä automaattinen ohjelmasynteesi on monien ohjelmistokehittäjien sekä suunnittelijoiden toiveiden täyttymys, mutta se jakaa myös paljon mielipiteitä. (Dakhel ym., 2022)

Copilot kehitettiin ensisijaisesti helpottamaan ohjelmistokehittäjien työtaakkaa sekä nopeuttamaan ohjelmistokehityksen prosessia, mutta siitä voivat hyötyä nykyään myös esimerkiksi harrastelijat ja opiskelijat. Copilot analysoi ja seuraa käyttäjän ohjelmakoodia ja antaa sen perusteella täydennys- ja korjausehdotuksia. Yksittäisten koodirivien lisäksi Copilot pystyy muodostamaan myös kokonaisia funktioita käyttäjän antamien ohjeiden eli komentojen perusteella erittäin nopeasti. Copilottia voi myös hyödyntää ohjelmakoodin ongelmien ratkaisuun ilman, että käyttäjän tarvitsisi erikseen etsiä ratkaisua esimerkiksi internetistä, mikä nopeuttaa työskentelyä huomattavasti. (Friedman, 2021)

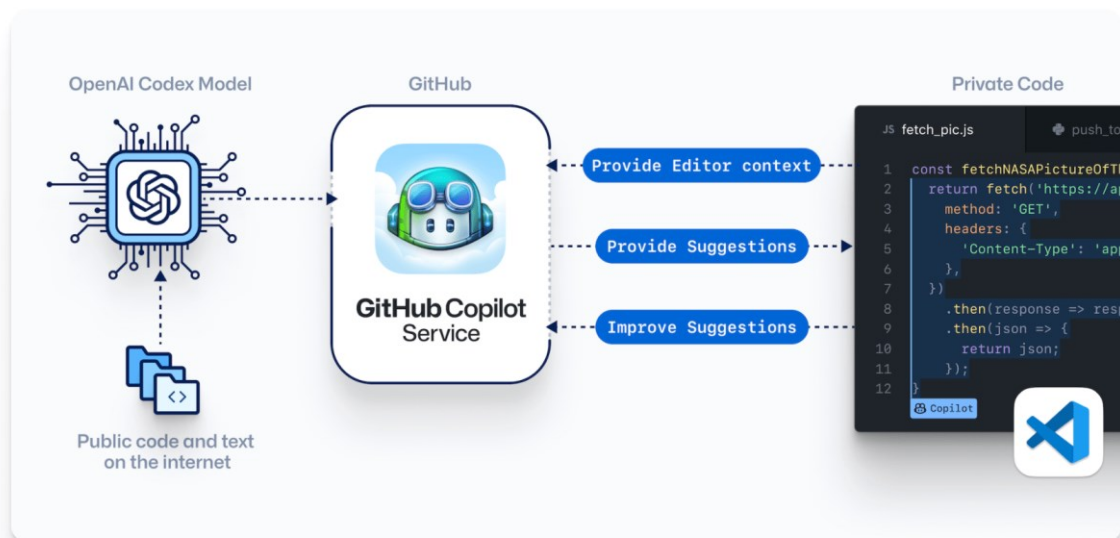
### 2.2.1 Copilotin kehitys ja toiminta

OpenAI on tekoälyn tutkimus- ja käyttöönottoyritys, jonka tavoitteena on varmistaa ja taata, että tekoäly hyödyntäisi vain yksittäisten tutkijoiden tai yritysten sijaan koko ihmiskuntaa. (OpenAI, 2015)

GitHubin Copilot saa virtansa Codexista, joka on OpenAI:n luoma uusi tekoälyjärjestelmä. Codex on GPT-3:sta hyödyntävä kielimalli (Generative Pretrained Transformer 3), joka sisältää jopa 12 miljardia parametria sekä 159 gigatavun edestä avointa lähdekoodia (Open Source), 54 miljoonasta GitHub-arkistosta. Codexia hyödyntävä Copilot pystyy generoimaan ohjelmakoodia useilla eri kielillä kuten TypeScriptillä, Pythonilla, Rubylla sekä JavaScriptillä, kun sille on annettu jokin konteksti eli kehote. Tällaisia kehoitteita ovat esimerkiksi kommentit, menetelmien nimet tai ympäröivä koodi. (Dakhel ym., 2022)

Kuvassa 3 on esitetty Copilotin toimintaperiaatetta ja sitä kuinka data liikkuu tekoälyohjelman sisällä. Copilotin automaattisesti luomat ehdotukset tulevat tiedoston kontekstista kuten funktioiden nimistä, koodikommenteista tai ympäröivästä koodista. Näiden perusteella Copilot tarjoaa ehdotuksia Codexista, joka on koulutettu valtavalla määrällä avointalähdekoodia. (Prokopets, 2021)

Kuva 3 Havainnollistava kuva Copilotin teknisestä toiminnasta (Prokopets, 2021)



Tällä hetkellä Copilot on mahdollista integroida jopa neljään eri tekstieditoriin, nämä ovat Visual Studio, Neovim, VS Code sekä JetBrains IDEs. Copilottia on myös mahdollista kokeilla 60 päivää ilmaiseksi, jonka jälkeen se kustantaa käyttäjälle 10 dollaria eli 9,2 euroa kuukaudessa tai vaihtoehtoisesti 100 dollaria eli 92 euroa vuosittaisella tilauksella, jolloin kaksi ensimmäistä kuukautta ovat ilmaisia. Copilotin tilaus edellyttää myös GitHubin käyttäjätiliä. (GitHub, 2022)

### 2.2.2 Copilotin hyödyt ja haitat

Koska tekoälyavusteinen ohjelmistokehitys on suhteellisen uusi ala GitHub suoritti vuonna 2021 tutkimuksen, missä haluttiin oppia laajasti Copilotin käyttökokemuksista yli 2000 ohjelmistokehittäjältä. Tutkimuksen tavoitteena oli keskittyä ennen kaikkea siihen, lisäksi Copilot kehittäjien tuottavuutta sekä näkevätkö kehittäjät muita etuja tehtävien nopeuttamisen lisäksi. Noin 60-75% vastaajista ilmoitti olevansa tyytyväisempi työnsä

laatuun ja tuntevansa vähemmän turhautumisen tunnetta ohjelmoidessaan. Vastaajista 73-87% oli sitä mieltä, että Copilot on auttanut säilyttämään työn mielenkiintoisena sekä lisäämään kärsivällisyyttä toistuvien tehtävien aikana. (Kalliamvakou, 2022)

Copilot on koulutettu kaikille julkisista arkistoista löytyville kielille, mikä lisää sen käytettävyyttä. Copilotin ehdottamien koodiratkaisujen laatu saattaa riippua käytetyn kielen harjoitusdatan määrästä ja monimuotoisuudesta. Esimerkiksi JavaScriptillä, joka on laajasti edustettuna julkisissa kirjastoissa ja yksi Copilotin parhaiten tuetuista kielistä, on suuremmat todennäköisyydet tuottaa laadukkaampia ratkaisuja kuin kielellä, josta on vähemmän dataa saatavilla. (GitHub, 2023)

Vaikka koneoppimisen myötä kehittynyt tekoäly on saanut suurta nostetta osakseen aiheuttaa se myös paljon huolta ohjelmistokehittäjien ja yritysten puolelta. Suurin huolenaihe Copilotin luoman koodin hyödyntämisessä on ehdottomasti tekijänoikeudet. Koska ei ole yleisessä tiedossa minkälaisella datalla Codexia hyödyntävä Copilot on koulutettu on käytännössä täysin mahdotonta varmuudella tietää, onko Copilotin suosittelema ratkaisu vapaata riistaa vai tekijänoikeuksilla suojattua koodia. Vaikka koneoppimisen ja datatieteen aloilla ei yleensä pidetä tekijänoikeusloukkauksena sitä, että käyttäjän syöte palautuu tulosteena, kuitenkin juridisessa mielessä tekijänoikeusrikkomukseen vaaditaan vain teoksen luvaton julkaiseminen tavalla, joka ei kuulu "reilun käytön" käsitteisiin. (Fitzpatrick, 2022)

Voisi olettaa, että koska Copilot ei voi luottaa reilun käytön käsitteeseen sen on täytynyt saada lupa kaikilta tekijänoikeuksien haltijoilta. GitHubin käyttöehtojen mukaan tallentamalla ohjelmistokoodin käyttämällä heidän palveluaan, loppukäyttäjä antaa GitHubille tiettyjä oikeuksia, jotka oikeuttavat esimerkiksi koodin kopiointiin ja sen tallentamiseen tietokantaan. Käyttöehdoissa kuitenkin edelleen todetaan, että lisenssi ei anna GitHubille oikeutta myydä, jakaa tai muuten käyttää sisältöä palvelun ulkopuolella. (Fitzpatrick, 2022)

## 2.3 ChatGPT

Vuonna 2022 Copilotinkin kehittämisessä mukana ollut OpenAI julkaisi ChatGPT-nimisen tekoälyohjelman. ChatGPT saavutti räjähdysmäisen suosion jo muutamassa kuukaudessa ja onkin siksi ollut laajasti mediassa ylistyksen sekä kritiikin kohteena. ChatGPT on yksinkertaisesti selaimesta löytyvä tekoälyä hyödyntävä keskustelubotti, joka vastaa kysymyksiin, ratkoo ongelmia ja pystyy kehittämään ohjelmistokoodia. (Pocock, 2023)

### 2.3.1 ChatGPT:n kehitys ja toiminta

Samoin kuin Copilotin Codex, myös ChatGPT on kehitetty GPT-3-kielimallia hyödyntäen. GPT-3:n avulla ChatGPT pystyy tuottamaan ihmisen kaltaista tekstiä ja se sisältää laajan valikoiman sovelluksia, jotka mahdollistavat kielenkääntämisen, kielenmallinnuksen sekä tekstin luomisen. Tähän mennessä GPT-3 onkin yksi suurimmista ja tehokkaimmista kielenkäsittelyä hyödyntävistä tekoälymalleista. (Hughes, 2023)

Vaikka ChatGPT ja GPT-3 ovat molemmat OpenAI:n kehittämiä kielenkäsittelymalleja, jotka perustuvat samaan teknologiaan, on niiden välillä kuitenkin jonkin verran eroja. GPT-3 on laajamittainen kielimalli, joka on koulutettu suurella määrällä tekstidataa kuten kirjoilla, artikkeleilla sekä verkkosivustoilla ja sitä käytetään laajalti erilaisiin tehtäviin kuten tekstin luomiseen, kielenkääntämiseen sekä kysymyksiin vastaamiseen. ChatGPT taas puolestaan on huomattavasti pienempi kielimalli, joka on koulutettu erityisesti vastausten luomiseen, joten se soveltuu paremmin tehtäviin, joissa mallin tulee tuottaa vastauksia keskustelun yhteydessä. ChatGPT sisältää 117 miljoonaa parametriä, minkä vuoksi se paljon kevyempi ja sitä voidaan hyödyntää useimmilla laitteilla verrattuna GPT-3:seen, joka sisältää jopa 175 miljardia parametriä. (Grincalaitis, 2022)

ChatGPT on koulutettu hyödyntämään valtavia määriä nimenomaan keskustelupohjaista dataa kuten nettikeskusteluita, minkä vuoksi sille voi antaa runsaasti erilaisia kehoitteita kuten kysymyksiä, kirjoituspyyntöjä ja kommentteja lähes mistä aiheesta tahansa, joiden perusteella se pystyy käymään erittäin ihmisenkaltaisia keskusteluja. ChatGPT:llä on myös niin sanottu ”muisti”, mikä mahdollistaa jatkokysymyksien esittämisen. Toisin kuin Copilot, joka on kehitetty varta vasten ohjelmistokehittäjien käyttöön ja koulutettu avoimella

lähdekoodilla, ChatGPT on chattibotti, joka kykenee vastaamaan lähes mihin tahansa kysymykseen millä tahansa kielellä. (Susnjak, 2022)

### 2.3.2 ChatGPT:n hyödyt ja haitat

Yksi suurimmista ChatGPT:n hyödyistä on se, että sitä ei ole varta vasten suunniteltu suorittamaan ainoastaan yhtä tehtävää vaan sitä voidaan soveltaa todella moneen eri käyttötarkoitukseen. ChatGPT:n taidot ulottuvat paljon pelkän tekstin tuottamista pidemmälle, sillä sitä voidaan soveltaa myös monimutkaisempiin tekstinkäyttöalueisiin kuten ohjelmointiin, kielikäännöksiin ja raporttien luomiseen. (Devaney, 2022)

ChatGPT:n käyttämisestä on tehty todella yksinkertaista, sillä se ei vaadi minkäänlaisia asennuksia ja sen käyttö on maksutonta. Lisäksi ohjelmalle on luotu erittäin yksinkertainen ja helppokäyttöinen käyttöliittymä, johon tallentuvat kaikki ChatGPT:n kanssa käydyt keskustelut. Tekoälybottiin pääsee käsiksi mistä tahansa selaimesta mutta koska sen suosio on ollut valtava monet ovat joutuneet pettymään yrittäessään päästä kokeilemaan sitä. Koska ChatGPT:n käyttäjiä on ollut samanaikaisesti enemmän kuin palvelimet pystyvät käsittelemään, on sisäänkirjautumista saattanut joutua odottelemaan tunneista päiviin. (Somoye, 2023)

Vaikka ChatGPT on osoittautunut helpoksi tavaksi kirjoittaa esimerkiksi esseitä, ohjelmoida tai saada nopeasti vastauksia mieltä askarruttaviin kysymyksiin, se ei kuitenkaan ole erehtymätön. Koska ohjelmiston ”älykkyyks” on täysin riippuvainen sille syötetystä datasta, joka voi olla peräisin esimerkiksi Wikipediasta, minne kuka tahansa pystyy kirjoittamaan valheellista tietoa, on ChatGPT:n tuottamia vastauksia on syytä tulkita kriittisesti. Virhemarginaalia lisää myös se, että ChatGPT saattaa tulkita käyttäjän kysymyksen väärin, mikäli se on muotoiltu huonosti tai epäselvästi. ChatGPT ei myöskään osaa vastata lähihistorian kysymyksiin koska maailmassa tiedon määrä kasvaa nopeammin kuin sen syöttäminen ChatGPT:lle. (Hughes, 2023)

OpenAI kertoo sivuillaan, että ChatGPT:n väärinkäytön ja haitallisen sisällön tuottamisen minimoimiseksi sen kehityksessä on käytetty moderointikirjastoa (Moderation API). Ideaalisesti ChatGPT:n pitäisi siis tunnistaa erittäin tarkasti, mikäli sille syötetty sisältö on

seksuaalista, vakivaltaista tai vihamielistä ja informoida käyttäjää vaarallisesta sisällöstä. Valitettavasti tästäkin huolimatta ChatGPT reagoi joskus haitallisiin ohjeisiin tai käyttäytyy puolueellisesti. (OpenAI, 2022)

## **2.4 Tekoälyn etiikka ja sen vaikutus opetukseen**

Tekoäly esittää tänä päivänä roolia miljardien ihmisten elämässä. Tekoäly voi esimerkiksi tarjota tukea miljoonille opiskelijoille mutta se kuitenkin aiheuttaa myös eettisiä kriisejä, kuten ihmisoikeusloukkauksia, ennakkoluuloja sekä syrjintää, jotka ovat huomion arvoisia. Jotta tekoäly voisi kehittyä tieteellisesti sekä rationaalisesti on välttämätöntä analysoida syvällisemmin sen aiheuttamia eettisiä ongelmia. (Li ym., 2021)

On selvää, että 2019 vuonna alkaneella Covid-19-pandemiolla on ollut valtava vaikutus oppilaitosten opetusmalleihin ja digitalisaatioon. Etäopetuksen yleistyessä rajoitusten myötä, siirrettiin opetus korkeakoulutasolla lähes kokonaan verkkoon, jonka seurauksena muutenkin yleistynyt digitalisaatio koulutuksen saralla kasvoi huippuunsa. Koska monta vuotta kestänyt rajoitusten aikakausi on jättänyt pysyvän jäljen nykypäivän opetusmetodeihin ja verkko-opetus on enemmänkin sääntö kuin poikkeus, on syytä pohtia, miten tekoälyohjelmien mahdollistamalta vilpiltä voitaisiin välttyä, jotta opetus pysyisi laadukkaana? (Sojonen, 2021)

Korkeakoulutuksen tulevaisuus on lujasti sidoksissa teknologian sekä uusien älykkäiden sovellusten kehitykseen. Tekoälyn kehitys avaa uusia mahdollisuuksia sekä haasteita korkeakoulujen opetusmetodeihin ja pakottaa korkeakouluja päivittämään sisäisiä opetusmenetelmiä sekä malleja. (Popenici & Kerr, 2017)

### **2.4.1 Tekoälyn eettiset ongelmat**

Tekoälyohjelmiin liittyy valitettavasti paljon eettisiä ongelmia kuten rasismia ja puolueellisuutta. Nämä ilmenevät tekoälysovelluksissa yleensä ennakkoluuloina, syrjintänä tai stereotyyppinä, joitakin tiettyjä ihmisryhmiä kohtaan, mikäli kielimallin kouluttamiseen käytetty data ei ole täysin puolueetonta. Tämän vuoksi esimerkiksi, jotkin ChatGPT:n antamat vastaukset saattavat olla epäoikeudenmukaisia. On myös mahdollista, että

kielimallia on koulutettu sellaisella datalla, joka edustaa vain murto-osaa väestöstä tietyissä kulttuureissa, joka poissulkee joitakin normeja. Tämä voi johtaa siihen, että tekoäly ei pysty ymmärtämään tai luomaan sisältöä ryhmille, jotka eivät ole edustettuina koulutustiedoissa. Monikielisyys on myös yksi epäkohta, joka ilmenee tekoälyohjelmissa. Usein kielimallien koulutusdata sisältää huomattavasti enemmän esimerkiksi englanninkielistä tekstiä, mikä johtaa siihen, että eri kielillä puhuvat ihmiset saattavat hyötyä vähemmän tekoälyohjelmien eduista. (Popenici & Kerr, 2017)

Yksi tekoälyohjelmien tuottama haaste on oppilaiden mahdollisuus hyödyntää tekoälyä opiskelussa. Esimerkiksi Copilot, joka on maksullinen apuväline mahdollistaa sen käytön ainoastaan niille oppilaille, joilla on siihen varaa, joka taas johtaa oppilaiden eriarvoisuuteen arviointiprosessissa. On erittäin vaikeaa erottaa opiskelijan omaa koodia Copilotin generoimasta koodista, joten Copilottia hyödyntävä opiskelija voi siis saada huomattavasti parempia arvosanoja ohjelmoinnin kurseista, joka on kohtuuton etu verrattuna opiskelijoihin, jotka eivät pysty Copilottia hyödyntämään. (Cotton ym., 2023)

Copilotin ja ChatGPT:n tuottamista vastauksista suurinta kohua ovat herättäneet tietosuojaongelmat ja plagioinnin mahdollisuus. On täysin mahdollista, että molempien tekoälyohjelmien kouluttamiseen on käytetty sellaista dataa, joka on tekijänoikeuksilla suojattua. Koska ChatGPT:n tai Copilotin sisältämä koulutusdata ei ole julkisessa tiedossa, on käytännössä täysin mahdotonta olla tietoinen siitä, ovatko tekoälyohjelmien generoimat vastaukset vapaasti käytettävissä. Myös vastauksiin viittaaminen oikeaoppisesti on erittäin hankalaa, koska ChatGPT ja Copilot eivät tarjoa käyttäjälle generoitujen vastausten alkuperää. (Fitzpatrick, 2022)

#### **2.4.2 Tekoälyohjelmien vaikutukset korkeakouluissa**

Vaikka tekoälyohjelmista voi olla paljonkin hyötyä oikein sovellettuna, asettaa niiden käyttö kuitenkin joitakin keskeisiä haasteita esimerkiksi kurssien arviointeihin. ChatGPT on suunniteltu luomaan esimerkiksi esseitä sille annettujen parametrien tai kehotusten perusteella. Tämä mahdollistaa järjestelmien hyödyntämisen huijaamiseen, sillä ChatGPT:llä tuotettuun kirjoitelmaan ei voi viitata sen tuntemattoman alkuperän vuoksi ja tällöin tekstin hyödyntäminen omana on plagiointia. Koska korkeakoulujen tarkoituksena on haastaa ja

kehittää oppilaiden omaa ajattelukykyä, on mahdollista, että tekoälyohjelmien yleistymisen saattaa lopulta johtaa tutkintojen devalvoitumiseen eli niiden arvon alenemiseen. (Cotton ym., 2023)

Miten tekoälyohjelmien tuomat vaikeudet arviointiprosesseissa tulisi siis ratkaista? On olemassa keinoja, joiden avulla opettajat voivat suunnitella kurssien sisältöä ja arviointiprosesseja niin, että ChatGPT:n tai Copilotin kaltaisten tekoälyohjelmien käyttöä voitaisiin minimoida. Yksi lähestymistapa olisi teettää opiskelijoilla tehtäviä, joissa osoitetaan kriittistä ajattelua, kommunikaatiotaitoja ja ongelmanratkaisukykyä sen sijaan, että opiskelijoita pyydetäisiin esimerkiksi koodaamaan yksinkertaisia ohjelmia tai kirjoittamaan esseitä tietyistä aiheista. Arviointikriteerinä voisi toimia esimerkiksi osallistuminen ryhmäkeskusteluihin tai muihin vuorovaikutteisiin menetelmiin, joissa arvioitaisiin oppilaiden tietojen ja taitojen soveltamiskykyä. (Cotton ym., 2023)

Lapin ammattikorkeakoulussa ChatGPT:n ominaisuuksiin suhtaudutaan positiivisesti ja se nähdään enemmänkin uutena työkaluna kuin uhkana. Digitaalisiin ratkaisuihin erikoistunut opettaja Petri Hannula kertoo, että opiskelijoille luodaan tehtäviä, joissa ChatGPT:n hyödyntäminen on sallittua esimerkiksi tiedon hakemiseen. Kehittämispäällikön Anu Pruikkosen mukaan tekoälyn väärinkäyttöä voidaan vähentää huomattavasti tehtävänantoja muokkaamalla. Kuitenkin Pruikkosen mukaan tekoälyohjelmien aiheuttama huoli on johtanut kansainvälisesti joissakin yliopistoissa teknologian hylkäämiseen esimerkiksi koetilanteissa, mikä siis käytännössä tarkoittaa, että tentit suoritetaan vain kynää ja paperia hyväksi käyttäen. Rakennusinsinööriopiskelijoiden Santeri Tepsan ja Oula Aikion mukaan tekoälyohjelmien hyödyntäminen mahdollisesti huijaustarkoitukseen on huolestuttavaa ja saattaa heikentää perusymmärrystä opintojen aiheista. (Rytkönen, 2023)

Tällä hetkellä tekoälyohjelmien kuten Copilotin ja ChatGPT:n antamiin vastauksiin tulee vielä suhtautua erittäin kriittisesti. Tampereen yliopiston professori Jyrki Nummenmaa kertoo, kuinka ChatGPT saattaa antaa hyvinkin itsevarmasti pätevältä kuulostavan vastauksen, joka ei todellisuudessa pidä paikkaansa. Tekoälysovelluksien älykkyyttä voi myös testata esittämällä niille kompakysymyksiä, johon tietää varmasti oikean vastauksen. (Pitkänen & Leponiemi, 2023)



Hämeenlinnan ammattikorkeakoulun tietojenkäsittelyn lehtori Tommi Saksan mukaan tekoälyohjelmat pystyvät generoimaan jopa pelottavan nopeasti täysin oikean ratkaisun yksinkertaiseen ohjelmointitehtävään, mikä mahdollistaa tehtävien suorittamisen kiitettävällä tasolla käytännössä ilman ymmärrystä tehtävästä tai sen vastauksesta. (Pitkänen & Leponiemi, 2023)

Hollantilainen dosentti Erdinç Saçan ymmärtää opettajien huolen tekoälyohjelmien väärinkäytöstä mutta ilmaisee myös, että pelko ei ole tarpeellista. Saçanin mukaan tekoälyn hyödyntämiseen opiskelussa ei tarvitse suhtautua negatiivisesti vaan enemmänkin uutena työkaluna. Se vaatii kuitenkin myös opetusmenetelmien päivittämistä vanhanaikaisista tehtävistä enemmänkin vuoropuhelutyylliseen opetukseen, missä oppilaiden kasvua ja oppimista voitaisiin mitata. Saçan kertoo myös, että esimerkiksi ICT-opiskelijat hyödyntävät jo ChatGPT:tä testauksessa ja ohjelmoinnin oppimisessa, joissa tekoäly toimii apuna haastavimmissa tehtävissä luokkakaverin tavoin. (Saçan, 2023)

## **2.5 Tekoälyn hyödyntäminen ohjelmistokehityksessä**

Tekoäly on mullistanut maailmaamme tavoilla, joita emme olisi voineet edes kuvitella muutamia vuosia sitten. Yritykset pyrkivät parantamaan tuottavuutta samalla pitäen kustannukset mahdollisimman alhaalla. Tekoäly voi auttaa tämän tavoitteen saavuttamisessa automatisoimalla vaarallisia ja toistuvia tehtäviä, jättäen ihmisille työvaiheet, joissa tekoälyä ei vielä pystytä hyödyntämään. Tekoälyn vaikutus näkyy jo kaikilla sektoreilla mutta erityisesti ohjelmistoteollisuudessa. (Parmar, 2022)

Ohjelmistokehitysprosessi on monimutkaista ja sitä pyritään jatkuvasti kehittämään virheettömämmäksi sekä yksinkertaisemmaksi. Uusien työkalujen ja teknologioiden kuten tekoälyn ansioista, ohjelmistokehittäjien on entistä helpompi tuottaa valmiita ohjelmistoja nopeammin. Tekoäly, jota käytetään yhdessä koneoppimisen ja syväoppimisen kanssa, voi nopeuttaa ja automatisoida ohjelmistojen testausta, jolloin ohjelmistotestaajat voivat käyttää aikaansa muihin toimintoihin. Toisin kuin manuaalinen testaus, tekoälypohjainen testaus eliminoi virheiden mahdollisuuden ja nopeuttaa ohjelmistokehitysprosessia, mikä takaa vaikuttavia tuloksia. (Parmar, 2022)

Tällä hetkellä tekoäly auttaa ohjelmistokehittäjiä kirjoittamaan tehokkaita koodeja. Tämä ei kuitenkaan tarkoita sitä, että tekoäly pystyisi korvaamaan ohjelmistokehittäjät, sillä se ei pysty kehittämään monimutkaisia koodeja yksinään. Ohjelmistokehityksessä käytettävät tekoälytyökalut voivat auttaa ohjelmoijia vähentämään toistuvia tehtäviä. Kun nämä tehtävät ovat poissa tieltä, koodaajat voivat keskittyä parantamaan taitojaan ja löytämään luovia tapoja tekoällyn hyödyntämiseen. Ihmisen älykkyyttä ei voi korvata. Vaikka tekoäly yksinkertaistaakin koodien kirjoittamista, se ei voi ajatella loogisesti eikä kykene myöskään tunneälyyn. Tekoäly ottaa valtavia harppauksia ohjelmistokehityksen uudistamiseksi, mutta näyttää erittäin epätodennäköiseltä, että se voisi korvata ihmiset kokonaan. (Parmar, 2022)

### 3 Tutkimuksen toteutus ja menetelmät

Tutkimuksen tarkoituksena on selvittää tekoälyohjelmien Copilotin sekä ChatGPT:n toiminnallisia eroavaisuuksia käytännössä. Tutkimustulokset ottavat kantaa siihen, kuinka näiden tekoälyohjelmien ominaisuuksia voitaisiin hyödyntää ohjelmoinnin opetuksessa niin, että ne eivät vaikuttaisi oppimiseen negatiivisesti. Tuloksia voidaan soveltaa tietojenkäsittelyopintojen tuleviin opetusmateriaaleihin sekä opetussuunnitelmaan.

Tutkimuksessa haluttiin myös selvittää ohjelmoinnin opettajien mielipiteitä sekä suhtautumista kehittyviin tekoälyjärjestelmiin. Tutkimustulokset kertovat kattavasti opettajien ajatuksista liittyen ChatGPT:n sekä Copilotin mahdollisuuksiin, riskeihin sekä ennen kaikkea riskien minimoimiseen, erityisesti ohjelmoinnin opetuksen näkökulmasta. Tulokset antavat esimakua siitä, kuinka opetusmalleja tullaan mahdollisesti jo lähiaikoina muokkaamaan teknologian kehittyessä.

Opinnäytetyön aineiston keräämisessä käytettiin menetelminä Copilotin ja ChatGPT:n vertailua sekä Hämeen ammattikorkeakoulun ohjelmoinnin opettajien teemahaastatteluja.

Opinnäytetyössä haluttiin vertailla kahta eniten kohua herättänyttä tekoälyohjelmaa ja tutkia, kumpi ohjelma olisi potentiaalisempi vaihtoehto hyödynnettäväksi työkaluna ohjelmoinnin opetuksessa. Tutkimuksessa ChatGPT:lle sekä Copilotille annetaan ratkaistavaksi kolme toisistaan eroavaa ohjelmointiin liittyvää tehtävää. Tekoälyohjelmien tuottamia vastauksia sekä niiden eroavaisuuksia analysoidaan tutkimuksen tuloksissa. Aikaisempi perehtyminen sekä Copilotin että ChatGPT:n toiminnallisuuksiin, hyötyihin sekä haittoihin opinnäytetyön tietoperustassa antoi suuntaa vertailututkimuksen tehtävien toteuttamiselle.

Teemahaastattelut pohjautuivat yhteensä 4 pääkysymykseen, joihin opettajat saivat vastata omin sanoin. Haastattelu suoritettiin kahdelle ohjelmoinnin opettajalle ja kysymyksillä haluttiin pääpiirteisesti selvittää, näkevätkö opettajat Copilotin tai ChatGPT:n uhkana ohjelmoinnin oppimiselle ja kuinka näitä ohjelmia voitaisiin hyödyntää ohjelmoinnin opetuksessa. Opettajien haastattelututkimuksien kysymysten laatimisessa sekä rajaamisessa

auttoivat tietoperustassa esiin tulleet opettajien ja oppilaiden mietteet tekoälyohjelmista, tekoälyohjelmien eettiset ongelmat sekä tekoälyohjelmien tekninen toiminta.

## 4 Copilotin ja ChatGPT:n eroavaisuuksien vertailu

Tekoälyohjelmat Copilot ja ChatGPT ovat jakaneet paljon mielipiteitä ja niitä hyödynnetään jo laajasti työelämässä. Tässä luvussa perehdytään siihen, kuinka Copilotin ja ChatGPT:n toimivat ja onko ohjelmien toiminnallisuuksien ja käytettävyyden välillä suuria eroavaisuuksia.

Tämä luku vastaa myös tutkimuskysymykseen ”Miten Copilot ja ChatGPT eroavat toisistaan?”. Eroavaisuuksien selvittämisen tarkoituksena on tutkia, kumpi tekoälyohjelma olisi ominaisuuksiltaan paremmin sovellettavissa ohjelmoinnin opetuksessa hyödyntämiseen sekä ohjelmoinnin oppimiseen.

Copilotin ja ChatGPT:n jo tiedossa olevia eroavaisuuksia ovat niiden hyvin erilaiset käyttöliittymät sekä ohjelmien tapa generoida koodia. Testauksessa sekä Copilotilta että ChatGPT:ltä kysyttävät kysymykset ovat identtisiä eikä niiden tuottamia vastauksia ole muokattu riippumatta siitä ovatko ne oikein. Koska Copilot ei ymmärrä suomenkielisiä kehoitteita toisin kuin ChatGPT, kaikki testauksessa esitetyt kysymykset on käännetty englanninkielisiksi.

Jotta eroavaisuuksia pystyttäisiin arvioimaan mahdollisimman tasapuolisesti, laajasti sekä laadukkaasti, pyritään testauksessa kehittämään kolme toisistaan eroavaa tehtävää, jotka annetaan Copilotilla ja ChatGPT:lle ratkaistavaksi. Tehtävien avulla pyritään kartoittamaan Copilotin ja ChatGPT:n kykyjä tuottaa koodia eri ohjelmointikielillä erilaisissa tehtävätyypeissä.

Ensimmäisen tehtävän ensimmäinen osa on yksinkertainen Java-ohjelma, jossa Copilotille ja ChatGPT:lle annetaan tehtäväksi generoida Lucky7-niminen peli, jossa ohjelman pitäisi arpoa käyttäjälle kolme lukua väliltä 1–10. Pelin ideana on, että jos yksi tai useampi arvotuista luvuista on 7 käyttäjälle ilmoitetaan, että hän on voittanut pelin. Mikäli yksikään arvotuista numeroista ei ole 7 käyttäjälle ilmoitetaan, että hän on hävinnyt.

Ensimmäisen tehtävän toisessa osassa pelin ohjelmakoodia jatketaan niin, että mukaan otetaan alkupanos. Käyttäjällä pitäisi siis pelin alussa olla 5 € rahaa ja jokaisen pelikierroksen hinta on 1 €. Tehtävän selostuksessa sanotaan, että mikäli ohjelma arpoo yhden 7, käyttäjä

voittaa kierroksesta 3 € ja mikäli kaksi 7 käyttäjä voittaa 5 € ja niin edelleen. Tehtävän avulla pyritään tutkimaan Copilotin ja ChatGPT:n kykyä generoida järkevä Java-ohjelma sekä sitä, kuinka ohjelmat suoriutuvat ohjelman jatkamisesta lisäohjeita annettaessa. Tehtävä on saatu tietojenkäsittelyn koulutuksen ohjelmoinnin perusteet-kurssimateriaalista ja se on suunniteltu ensimmäisen vuoden opiskelijoille.

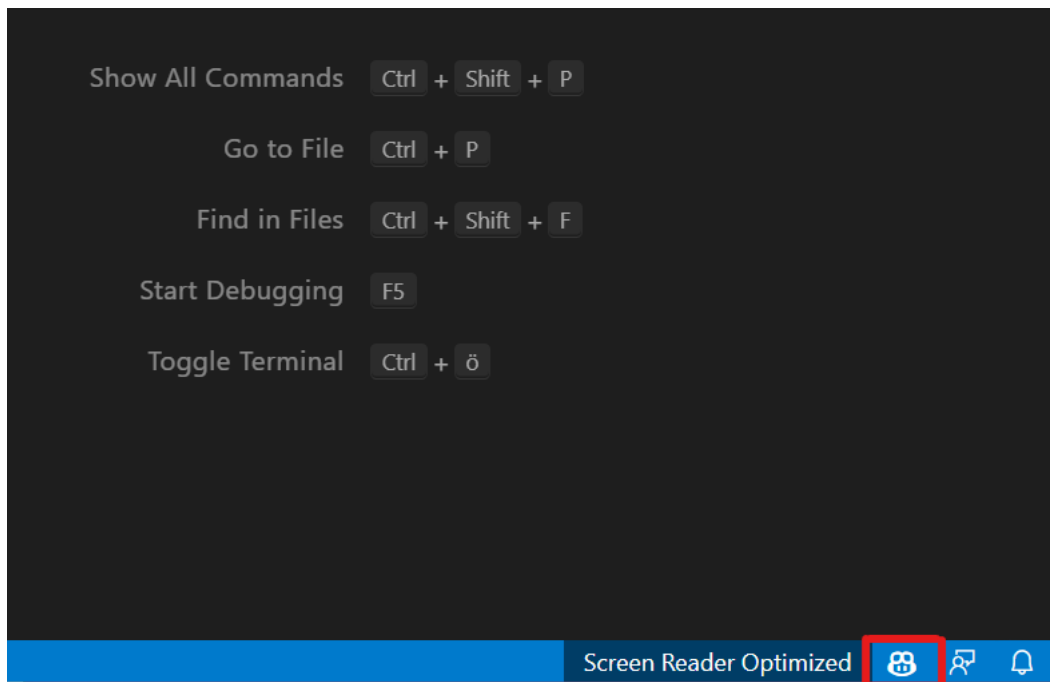
Ohjelmistotestaus on yksi tietojenkäsittelykoulutuksen osa-alue, jota opetetaan Hämeen ammattikorkeakoulussa toisen vuoden tietojenkäsittely opiskelijoille. Toisessa tehtävässä tutkittiin Copilotin sekä ChatGPT:n kykyä generoida neljä yksikkötestitapausta yksikertaiselle laskinohjelmalle. Molemmille tekoälyohjelmille annettiin valmiiksi koodattu laskinohjelma, jonka perusteella testitapaukset pyydettiin luomaan. Tehtävässä mitattiin Copilotin ja ChatGPT:n taitoja luoda vastauksia ympäröivän kontekstin perusteella. Toisin kuin ensimmäisessä vertailussa, tässä tehtävässä käytettiin ohjelmointikielenä Pythonia.

Kolmannessa tehtävässä haluttiin tutkia Copilotin ja ChatGPT:n ongelmanratkaisutaitoja. Tehtävä koostuu lyhyestä JavaScript-koodista, jonka tarkoituksena on luoda eri automerkeistä koostuva joukko (Array) ja tulostaa konsoliin kaikki sille asetetut automerkit. Tekoälyohjelmille syötettyyn koodiin oli jätetty yhteensä neljä erilaista virhettä, ja tehtävän tarkoituksena olikin selvittää, kuinka monta virhettä Copilot sekä ChatGPT pystyvät ohjelmakoodista löytämään ja korjaamaan.

#### **4.1 Ohjelmointitehtävien suorittaminen Copilotilla**

Copilotin testauksen kehitysympäristönä hyödynnettiin Visual Studio Codea, johon asennettiin Github Copilot lisäosa. Copilotin asennukseen vaaditaan GitHubin käyttäjätili sekä Copilotin kuukausittainen tai vuosittainen tilaus. Copilottia pystyy kuitenkin kokeilemaan 60 päivää ilmaiseksi. Kun Copilot on asennettu, se aktivoituu automaattisesti Visual Studio Codessa ja se on mahdollista deaktivoida Kuvassa 4, oikeassa alakulmassa näkyvästä Copilot-ikonista.

Kuva 4 Copilotin aktivointi ja deaktivointi



Copilotin ollessa aktiivinen se tuottaa jatkuvasti täydennys ehdotuksia käyttäjän kirjoittaessa ohjelmakoodia. Kuitenkin tässä testitapauksessa Copilotin toiminnallisuutta testataan antamalla sille komentoja eli englanninkielisiä kommentteja tekstieditoriin, joiden perusteella se generoi ohjelmakoodia. Tehtävän luonteen mukaan komento voidaan syöttää joko kokonaisuena tai jaettuna pienempiin osiin, mikäli tehtävä on laaja. Se, miten komentoja pystyy syöttämään Copilotille, on riippuvainen käytettävästä ohjelmointikielestä. Esimerkiksi Java-kielellä kommentteja pystyy syöttämään Copilotille kahdella tavalla. Ensimmäinen tapa on kirjoittaa tekstieditoriin kaksi kauttamerkkiä `"/"`, joiden perään kirjoitetaan selkokieline komento. Tätä metodia kutsutaan *"rivikommentiksi"* (Inline Comment) ja sitä käytetään, kun kommentti mahtuu yhdelle riville. Toinen tapa on luoda niin kutsuttu *"lohkokommentti"* (Block Comment) merkeillä `/*-*/`, joiden väliin voi kirjoittaa pidemmän kommentin, joka on useamman rivin pituinen.

Kuva 5 Komennon syöttäminen Copilotille Java-kielellä

```
1 public class comments {
2
3     //Insert the comment here: This is an Inline Comment
4
5
6
7     /*
8     | Inset the comment here: This is a Block Comment
9     */
10 }
11
```

Copilotin generoimat koodiehdotukset ilmestyvät editoriin harmaina ja mikäli ehdotus on käyttäjän mielestä hyvä, sen voi hyväksyä painamalla Tab-näppäintä. Enteriä painamalla ehdotuksen voi hylätä. Copilot pystyy tuottamaan myös useampia vaihtoehtoja pulmasta riippuen. Kuvassa 6 havainnollistetaan, miten muita ohjelman generoimia vaihtoehtoja pystyy tarkastelemaan esimerkiksi viemällä hiiren kursorin harmaanväriseen ehdotuksen päälle ja auki ilmestyvästä valikosta painamalla joko "Open GitHub Copilot (Ctrl + Enter)", jolloin vasemmalle aukeavasta ikkunasta käyttäjä näkee kaikki saatavilla olevat vaihtoehdot tai selaamalla ehdotuksia eteen tai taaksepäin näppäimistä "Next (Alt + >)" ja "Previous (Alt + <)"



Kuva 6 Copilotin generoiman koodin hyväksyminen ja valinta

```

1 public class alternative {
2     Suggestion:
3     for (int i = 0; i < list.length; i++) {
4         list[i] = (int) (Math.random() * 100);
5     }
6     //
7     for (int i = 0; i < list.length; i++) {
8         list[i] = (int) (Math.random() * 100);
9     }
10
11 }

```

```

1 Synthesizing 10/10 solutions
2
3 =====
4
5     Accept Solution
6     for (int i = 0; i < list.length; i++) {
7         list[i] = (int) (Math.random() * 100);
8     }
9
10    //Print the list
11    for (int i = 0; i < list.length; i++) {
12        System.out.print(list[i] + " ");
13    }
14
15    //Find the first odd number
16    int result = 0;
17    for (int i = 0; i < list.length; i++) {
18        if (list[i] % 2 == 1) {
19            result = list[i];
20            break;
21        }
22    }
23
24    //Print the result
25    System.out.println();
26    System.out.println("The first odd number is: " + result);
27
28    =====
29
30    Accept Solution
31    for (int i = 0; i < list.length; i++) {
32        list[i] = (int) (Math.random() * 100);
33    }
34
35    //Print the list
36    for (int i = 0; i < list.length; i++) {
37        System.out.print(list[i] + " ");
38    }

```

#### 4.1.1 Ensimmäinen testitapaus: Lucky7

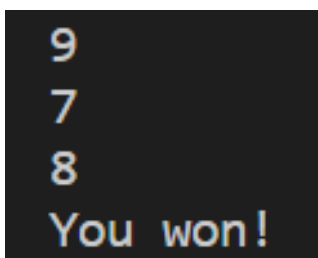
Ensimmäisestä tehtävästä Copilot suoriutui erittäin nopeasti kohtalaisella tuloksella. Koska tehtävän anto oli kohtuullisen pitkä, se jaettiin kahteen osaan. Tehtävän jaottelun tarkoituksena oli myös tutkia kuinka Copilot suoriutuu tehtävän jatkamisesta. Ohjelmakoodi 1 havainnollistaa, kuinka Copilot tuotti kahden rivikommentin perusteella satunnaisten numeroiden generoinnin välillä 1-10, kyseisten numeroiden tulostuksen terminaaliin sekä numeroiden tarkastamisen, jonka perusteella käyttäjälle kerrotaan, onko hän voittanut vai hävinnyt pelin.

## Ohjelmakoodi 1 Copilotin generoima koodi: Lucky7 osa 1

```
1 public class App {
   Run | Debug
2     public static void main(String[] args) throws Exception {
3
4         //Make a game that draws 3 numbers between 1-10 and prints the numbers.
5         //If any of the numbers is a seven, tell the user you won. If nothing is a seven, tell the user you lost.
6
7         int[] numbers = new int[3];
8         for (int i = 0; i < numbers.length; i++) {
9             numbers[i] = (int)(Math.random() * 10) + 1;
10        }
11
12        for (int i = 0; i < numbers.length; i++) {
13            System.out.println(numbers[i]);
14        }
15
16        for (int i = 0; i < numbers.length; i++) {
17            if (numbers[i] == 7) {
18                System.out.println(x: "You won!");
19                break;
20            }
21            else if (i == numbers.length - 1) {
22                System.out.println(x: "You lost!");
23            }
24        }
25    }
```

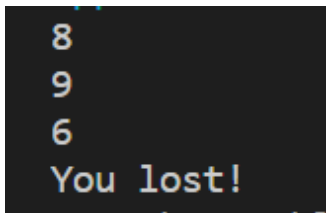
Tehtävän ensimmäisen osion suorittaminen sujui Copilotilta erittäin nopeasti ja täysin ongelmitta. Tehtävän yksinkertaisuuden vuoksi, sen pystyy suorittamaan täysin kiitettävästi Copilotin avulla, käytännössä ilman minkäänlaista ymmärrystä siitä, mitä koodissa tapahtuu. Kuvissa 7 ja 8 on esitetty ohjelman antamat tulokset. Ohjelma ilmoittaa käyttäjälle, että tämä on voittanut, mikäli yksi tai useampi numeroista on 7. Jos yksikään kolmesta numerosta ei ole 7 ohjelma ilmoittaa, että käyttäjä on hävinnyt.

Kuva 7 Copilot Lucky7 osa 1 pelin voitto



```
9
7
8
You won!
```

Kuva 8 Copilot Lucky7 osa 1 pelin häviö



Tehtävän toisessa osiossa Copilottia pyydettiin jatkamaan peliä niin, että pelikierrokset maksavat euron ja mikäli ohjelma arpoo yhden tai useamman 7 käyttäjä voittaa siitä rahaa. Ohjelmakoodi 2 havainnollistaa, kuinka Copilot lisää rahan peliin, luo ehtolauseet voittosummille ja lopuksi tulostaa, kuinka paljon käyttäjällä on rahaa jäljellä pelin päättyessä. Ohjelma myös miinustaa aina yhden euron käyttäjän rahasummasta kierroksen lopussa.

Ohjelmakoodi 2 Copilotin generoima koodi: Lucky7 osa 2

```

25
26 //Add money to the game. The user has 5 euros money at the beginning.
27 //If one of the drawn numbers is seven, the player wins 3 euros.
28 //If two numbers are seven, the user wins 5 euros and so on.
29 //Each round costs one euro.
30 //Print the amount of money the user has left after the game.
31
32     int money = 5;
33     int sevenCount = 0;
34     for (int i = 0; i < numbers.length; i++) {
35         if (numbers[i] == 7) {
36             sevenCount++;
37         }
38     }
39
40     if (sevenCount == 1) {
41         money += 3;
42     }
43     else if (sevenCount == 2) {
44         money += 5;
45     }
46     else if (sevenCount == 3) {
47         money += 10;
48     }
49
50     money -= 1;
51     System.out.println("You have " + money + " euros left.");
52
53 }
54 }
55

```

Vaikka Copilot suoriutui testistä täysin moitteettomasti tehtävänannon mukaisesti, on kuitenkin huomioitava, että ohjelma on tuollaisenaan vielä melko vajavainen. Koska ohjelmasta puuttuu täysin toistorakenne eli ”silmukka”, pelin pystyy pelaamaan ainoastaan kerran, jonka jälkeen ohjelma on ajettava uudestaan. Tämä tarkoittaa sitä, että loppurahasumma ei voi laskea alle neljän euron ja suurin voittosumma voi olla korkeintaan 14 euroa, mikäli käyttäjä saa kolme 7 samaan aikaan. Copilot ei siis oma-aloitteisesti luonut peliin toistorakennetta, jotta peliä pystyisi jatkamaan yhden kierroksen jälkeen.

Kuvassa 9 on esitetty ohjelman vastaus terminaalissa, mikäli käyttäjä on voittanut pelin. Käyttäjän rahapottiin lisätään voittosumma riippuen siitä, kuinka monta seiskaa ohjelma on arponut. Tässä esimerkissä käyttäjälle arvottiin kaksi 7, minkä vuoksi voittosumma on 5 euroa. Tämän jälkeen käyttäjän rahasummasta vähennetään 1 euron kierrosmaksu. Kuva 10 havainnollistaa ohjelman vastauksen, kun käyttäjä on hävinnyt pelin. Tällöin käyttäjän viiden euron aloitusrahasta vain vähennetään 1 euron kierrosmaksu.

Kuva 9 Copilot Lucky7 osa 2 pelin voitto

```
7
3
7
You won!
You have 9 euros left.
```

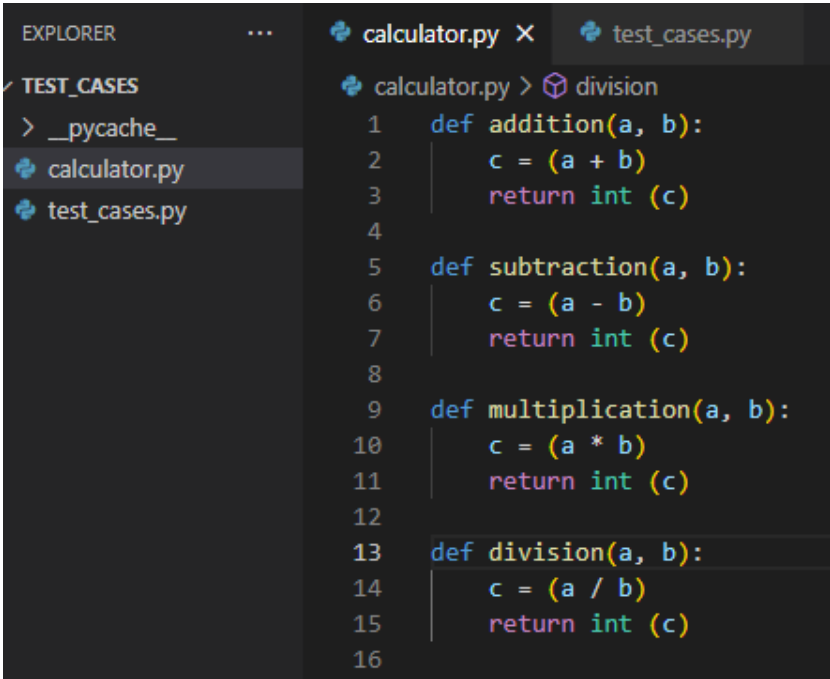
Kuva 10 Copilot Lucky7 osa 2 pelin häviö

```
4
1
6
You lost!
You have 4 euros left.
```

#### 4.1.2 Toinen testitapaus: Yksikkötestitapausten generointi

Toisessa tehtävässä Copilotille annettiin valmiiksi ohjelmoitu koodi yksinkertaisesta laskinohjelmasta. Ohjelmakoodi 3 on esitetty koodi, minkä perusteella Copilottia pyydettiin luomaan neljä yksikkötestitapausta. Tässä tehtävässä varsinainen ohjelmakoodi on eri tiedostossa kuin tehtävänanto ja yksikkötestien generointi.

Ohjelmakoodi 3 Laskin-ohjelman syöttö Copilotille



```
EXPLORER  ...  calculator.py X  test_cases.py
TEST_CASES
  > __pycache__
  calculator.py
  test_cases.py
calculator.py > division
1  def addition(a, b):
2      c = (a + b)
3      return int(c)
4
5  def subtraction(a, b):
6      c = (a - b)
7      return int(c)
8
9  def multiplication(a, b):
10     c = (a * b)
11     return int(c)
12
13  def division(a, b):
14     c = (a / b)
15     return int(c)
16
```

Copilotilta kului vain sekunteja testitapausten luomiseen. Ohjelmakoodissa 4 on havainnollistettu Copilotille syötettyä komentoa sekä sen generoimaa vastausta. Copilot onnistui myös generoimaan kaikki muut oleelliset osat testien suorittamista varten kuten importit sekä luokan luomisen, varsinaisten yksikkötestien lisäksi. Copilot ei kuitenkaan ottanut yhden testitapausten luomisessa huomioon annetun laskinohjelman c-muuttujaa, jonka tulisi olla integer eli kokonaisluku. Tämän takia viimeinen yksikkötesti "division" ei ole oikein, sillä vastaus ei palauta kokonaislukua vaan murtoluvun. Ongelma olisi kuitenkin

helppo ratkaista, joko muuttamalla c-muuttuja float tyyppiseksi tai vaihtamalla viimeisen testitapauksen muuttujien arvoja niin, että niiden jakolaskusta tulisi kokonaisluku.

#### Ohjelmakoodi 4 Copilotin generoimat yksikkötestitapaukset

```

test_cases.py > TestCalculator > test_subtraction
1  import unittest
2  import calculator
3
4  #Create 4 unit test cases for the calculator including arrange, act, and assert
5
6  class TestCalculator(unittest.TestCase):
7
8      def test_addition(self):
9          #Arrange
10         num1 = 2
11         num2 = 3
12         #Act
13         result = calculator.addition(num1, num2)
14         #Assert
15         self.assertEqual(result, 5)
16
17     def test_subtraction(self):
18         #Arrange
19         num1 = 2
20         num2 = 3
21         #Act
22         result = calculator.subtraction(num1, num2)
23         #Assert
24         self.assertEqual(result, -1)
25
26     def test_multiplication(self):
27         #Arrange
28         num1 = 2
29         num2 = 3
30         #Act
31         result = calculator.multiplication(num1, num2)
32         #Assert
33         self.assertEqual(result, 6)
34
35     def test_division(self):
36         #Arrange
37         num1 = 2
38         num2 = 3
39         #Act
40         result = calculator.division(num1, num2)
41         #Assert
42         self.assertEqual(result, 0.6666666666666666)
43
44     if __name__ == '__main__':
45         unittest.main()

```

#### 4.1.3 Kolmas testitapaus: Ongelmanratkaisu

Kolmannessa tehtävässä Copilotille syötettiin Ohjelmakoodissa 5 näkyvä, valmiiksi väärin ohjelmoitu JavaScript-koodi sekä tehtävänanto rivikommentein. Copilotilta kysyttiin, mitä

vikaa koodissa on ja kuinka sen voisi korjata. Koodissa esiintyvät virheet ovat hyvin pieniä ja yksinkertaisia.

#### Ohjelmakoodi 5 Virheellisen ohjelmakoodin syöttö Copilotille

```
1
2 //What is wrong with this code?
3 //How can I fix it?
4
5 const cars=();
6 car[0]="Volvo";
7 car[1]="BMW";
8 car[2]="Saab";
9 car[120]="Ford";
10 car[101]="Fiat";
11
12 for (let i=0;i<cars.length;i+){
13     consolelog(cars[i]);
14 }
15
```

Copilot onnistui löytämään ja korjaamaan väärin ohjelmoidusta koodista kolme neljästä virheestä. Ohjelmakoodi 6 havainnollistaa, mitkä virheet Copilot on löytänyt koodista ja kuinka ne tulisi korjata. Ensimmäinen virhe, jonka Copilot havaitsi, oli joukon luomisessa käytetyt väärät sulkeet. Joukon luomisessa tulisi käyttää hakasulkeita tavallisten sulkeiden sijaan. Copilot löysi koodista myös kaksi muuta puutteellista kohtaa, jotka se on tulkinnut kirjoitusvirheiksi. Copilot huomauttaa, että for-loopista puuttuu merkki "+" ja joukon tulostuksesta "consolelog" puuttuu piste sanojen välistä. Copilot ei kuitenkaan löytänyt neljättä virhettä ohjelmakoodista, vaikka se on alleviivattu selkeästi punaisella.

## Ohjelmakoodi 6 Copilotin löytämät virheet sekä ratkaisuehdotukset

```

1
2 //What is wrong with this code?
3 //How can I fix it?
4
5 const cars=();
6 car[0]="Volvo";
7 car[1]="BMW";
8 car[2]="Saab";
9 car[120]="Ford";
10 car[101]="Fiat";
11
12 for (let i=0;i<cars.length;i+){
13 |   console.log(cars[i]);
14 }
15
16 You have a typo in the first line of code:
17
18 const cars=();
19
20 should be:
21
22 const cars=[];
23
24 When you initialize an array, you need to use square brackets, not parentheses.
25
26 Also, you have a typo in your for loop:
27
28 for (let i=0;i<cars.length;i+){
29
30 should be:
31
32 for (let i=0;i<cars.length;i++){
33
34 The for loop is missing a curly bracket at the end.
35
36 And the console.log() should be:
37
38 console.log(cars[i]);
39

```

### 4.2 Ohjelmointitehtävien suorittaminen ChatGPT:llä

ChatGPT:n testausta varten ei tarvitse tehdä erillisiä asennuksia tai tilausta, sillä siihen pääsee käsiksi selaimista toistaiseksi vielä täysin ilmaiseksi. Ensimmäistä kertaa ChatGPT:tä käytettäessä on kuitenkin luotava käyttäjätili voimassa olevaa sähköpostia ja salasanaa käyttäen.

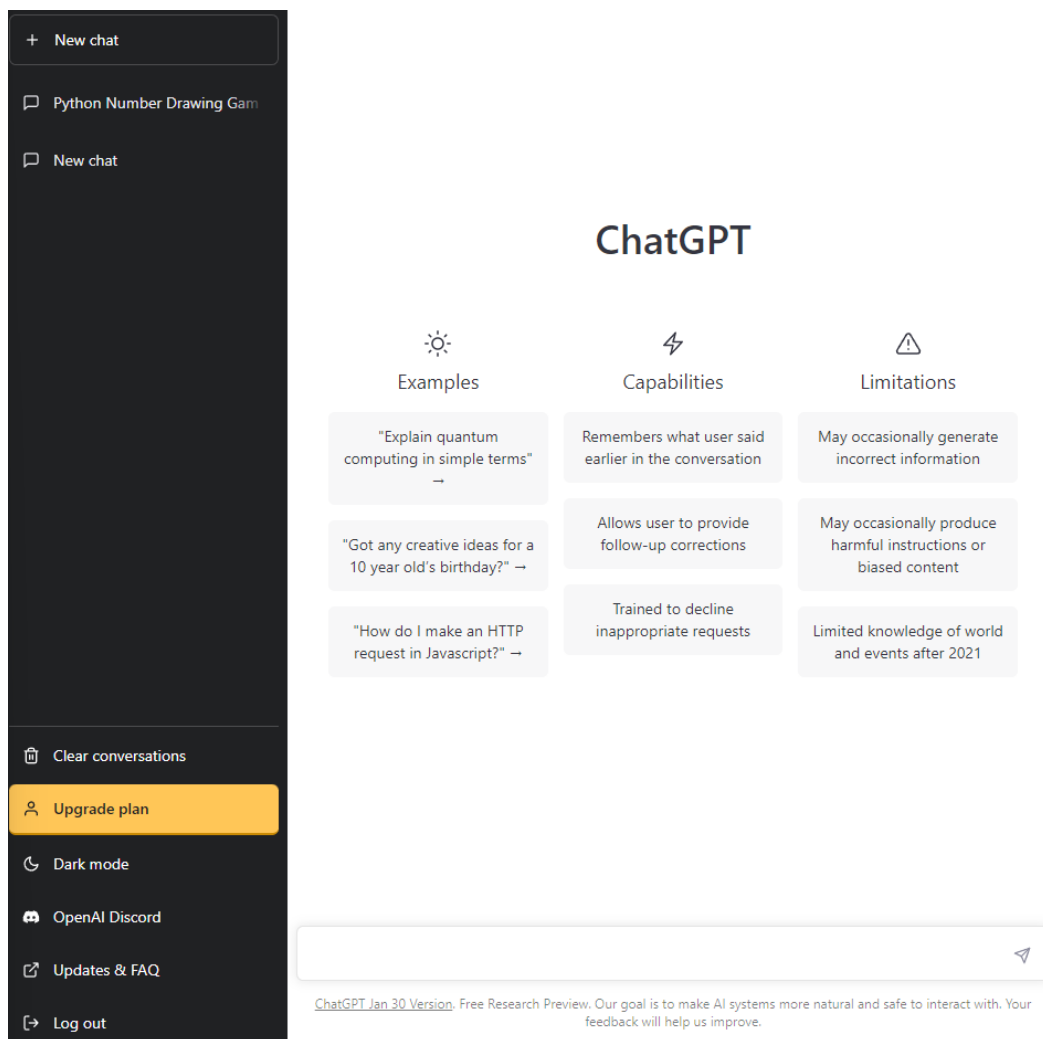


Sisäänkirjautumisen jälkeen ChatGPT ikkunan alareunassa näkyvään tekstikenttään pystyy kirjoittamaan komennon tai kysymyksen, johon ohjelma vastaa. ChatGPT myös tallentaa kaikki sen kanssa käydyt keskustelut sivun vasempaan reunaan, jotta niihin voi halutessaan palata myöhemmin. Uuden keskustelun pystyy aloittamaan sivun yläreunassa näkyvästä "New Chat" painikkeesta.

ChatGPT:n etusivun avautuessa ohjelma antaa käyttäjälle esimerkkiehdotuksia kysymyksistä, joita siltä voi kysyä. ChatGPT kertoo myös avautumisenäkymässä sen vahvuuksista, joita ovat esimerkiksi ChatGPT:n kyky muistaa, mitä käyttäjä on sanonut keskustelussa aiemmin ja sen kyky hylätä epäsovikat pyynnöt koulutuksen ansioista. ChatGPT myös varoittaa käyttäjää sen heikkouksista kuten siitä, että se saattaa toisinaan tuottaa väärää tietoa ja että sen tietoperusta maailmasta rajoittuu vain vuoteen 2021 asti.

ChatGPT:n tapa generoida koodia on hyvin erilainen kuin Copilotin. ChatGPT suorittaa pyynnön loppuun antamatta käyttäjälle erilaisia vaihtoehtoja. Vasta pyynnön suorituksen jälkeen käyttäjä pystyy esittämään tarkempia kysymyksiä tai pyytämään ChatGPT:tä muokkaamaan generoitua koodia haluamaansa suuntaan. Mutta koska ChatGPT muistaa sen kanssa käydyt keskustelut se pystyy jatkamaan ja muokkaamaan jo aikaisemmin luotua ohjelmakoodia.

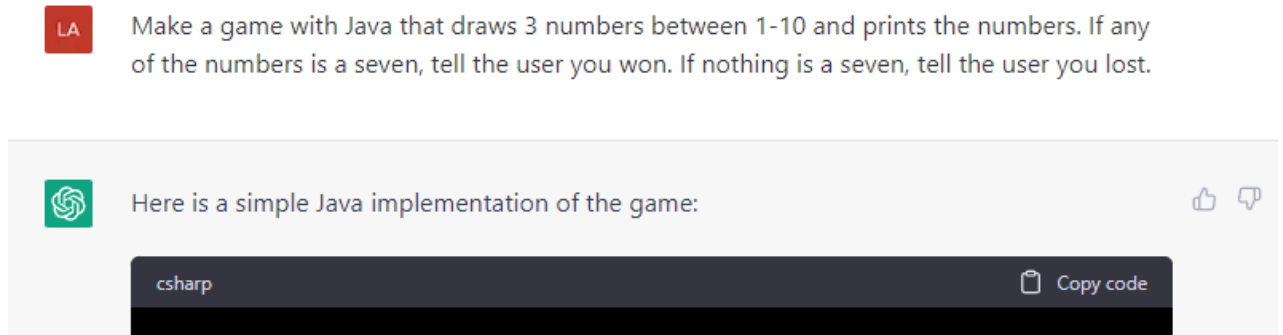
Kuva 11 ChatGPT:n käyttöliittymä



#### 4.2.1 Ensimmäinen testitapaus: Lucky7

Samoin kuin Copilot, ChatGPT suoriutui tehtävän ensimmäisestä osasta täysin ongelmitta ja halutulla tavalla. Tekoälyohjelmien generoimat vastaukset eivät kuitenkaan ole samanlaisia, vaikka niille annetut komennot ovat täysin identtiset. Kuvassa 12 on esitetty tehtävän ensimmäisen osion komennot syötettynä ChatGPT:lle.

Kuva 12 Lucky7 osa 1: Komennon syöttö ChatGPT:lle



Copilotin tavoin, ChatGPT onnistui generoimaan paikkaansa pitävän vastauksen vain kymmenissä sekunneissa. Ohjelmakoodi 3 havainnollistaa ChatGPT:n luomaa ohjelmakoodiversiota Lucky7-pelin ensimmäisestä osasta. Koodista voidaan havaita, että ChatGPT:n generoimassa koodissa on muutamia eroavaisuuksia verrattuna Copilotin hieman yksinkertaisempaan koodiin. ChatGPT käyttää voiton tarkistamiseen boolean-muuttujaa, kun taas Copilot hyödynsi integeriä eli kokonaislukuja. ChatGPT myös lisäsi tulostukseen informoivan tekstin käyttäjälle, jossa sanotaan ”Numerot ovat: (The numbers are:)” vaikka ohjeistuksessa ei asiasta erikseen mainita.

## Ohjelmakoodi 7 Lucky7 osa 1: ChatGPT:n generoima koodi

```
1  import java.util.Random;
2
3  public class chat2 {
4      Run | Debug
5      public static void main(String[] args) {
6          Random rand = new Random();
7
8          int[] numbers = new int[3];
9          for (int i = 0; i < 3; i++) {
10             numbers[i] = rand.nextInt(bound: 10) + 1;
11         }
12
13         System.out.print(s: "The numbers are: ");
14         for (int i = 0; i < 3; i++) {
15             System.out.print(numbers[i] + " ");
16         }
17         System.out.println();
18
19         boolean win = false;
20         for (int i = 0; i < 3; i++) {
21             if (numbers[i] == 7) {
22                 win = true;
23                 break;
24             }
25         }
26
27         if (win) {
28             System.out.println(x: "You won!");
29         } else {
30             System.out.println(x: "You lost.");
31         }
32     }
}
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

The numbers are: 7 7 10  
You won!

The numbers are: 1 6 6  
You lost.

Seuraavaksi ChatGPT:tä pyydettiin jatkamaan koodia niin, että siinä käytetään rahaa.

Ohjeistus annettiin ChatGPT:lle täysin samoja sanoja käyttäen kuin Copilotinkin

ohjeistuksessa mutta vain yhteen kirjoitettuna ja pisteillä erotettuna. Kuvassa 13 on esitetty pyynnön syöttäminen sekä ChatGPT:n vastaus.

Kuva 13 Lucky7 osa 2: Komennon syöttö ChatGPT:lle



Add money to the game. The user has 5 euros money at the beginning. If one of the drawn numbers is seven, the player wins 3 euros. If two numbers are seven, the user wins 5 euros and so on. Each round costs one euro. Print the amount of money the user has left after the game.



Here is a simple Java implementation of the game with money:



ChatGPT:n generoima vastaus tehtävän toiseen osaan poikkeaa huomattavasti Copilotin tuottamasta koodista. Kuten Ohjelmakoodi 4:ssä on havainnollistettu, ChatGPT onnistui luomaan peliin toistorakenteen ilman erillistä pyyntöä, jotta peliä pystyy jatkamaan painamalla Enter-näppäintä. Peliä pystyy siis pelaamaan niin kauan kun käyttäjällä on rahaa jäljellä, toisin kuin Copilotin luomassa pelissä, jossa ohjelma on ajettava aina uudelleen.

Niin kuin tehtävän ensimmäisessäkin osassa, ChatGPT lisäsi ohjelmakoodin tulostukseen käyttäjälle hyödyllisiä selitteitä, ilman erillistä pyyntöä. Tulostuksessa eritellään käyttäjälle rivi riviltä, kuinka peli etenee ja miten peliä pystyy jatkamaan. Ohjelmakoodi myös kertoo käyttäjälle, mikäli rahat ovat loppuneet ja peli päättyy.

ChatGPT ei kuitenkaan täysin onnistunut luomaan pelin toista osaa tehtävänannon mukaisesti. Tehtävässä on eritelty, kuinka paljon rahaa käyttäjän tulisi voittaa, kutakin arvottua 7 numeroa kohden. Ohjelmakoodi 4 havainnollistaa, kuinka käyttäjä voittaa euron, pelin arpoessa yhden 7 numeron, vaikka tehtävänannon mukaisesti käyttäjän tulisi tässä tapauksessa voittaa 3 euroa.

## Ohjelmakoodi 8 Lucky7 osa 2: ChatGPT:n generoima koodi

```

3 public class chat {
4     public static void main(String[] args) {
5         Random rand = new Random();
6         int money = 5;
7
8         while (money > 0) {
9             System.out.println("You have " + money + " euros. Each round costs 1 euro.");
10            System.out.println(x: "Press enter to play.");
11            try {
12                System.in.read();
13            } catch (Exception e) {}
14            int[] numbers = new int[3];
15            for (int i = 0; i < 3; i++) {
16                numbers[i] = rand.nextInt(bound: 10) + 1;
17            }
18            System.out.print(s: "The numbers are: ");
19            for (int i = 0; i < 3; i++) {
20                System.out.print(numbers[i] + " ");
21            }
22            System.out.println();
23            int win = 0;
24            for (int i = 0; i < 3; i++) {
25                if (numbers[i] == 7) {
26                    win++;
27                }
28            }
29            money -= 1;
30            money += win;
31            System.out.println("You won " + win + " euros.");
32            System.out.println("You have " + money + " euros left.");
33        }
34        System.out.println(x: "You have no money left. Game over.");
35    }

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

```

You have 5 euros. Each round costs 1 euro.
Press enter to play.


The numbers are: 1 5 4
You won 0 euros.
You have 4 euros left.
You have 4 euros. Each round costs 1 euro.
Press enter to play.
The numbers are: 5 7 9
You won 1 euros.
You have 4 euros left.
You have 4 euros. Each round costs 1 euro.
Press enter to play.

```

#### 4.2.2 Toinen testitapaus: Yksikkötestitapausten generointi

Toisessa tehtävässä ChatGPT:lle syötettiin tehtävänannon lisäksi myös siihen liittyvä yksinkertainen laskinohjelman koodi. Kuvassa 14 on esitetty komennon ja ohjelmakoodin syöttö ChatGPT:lle ja niin kuin aikaisemmassakin tehtävässä, tehtävänanto sekä siihen liittyvä koodi on identtinen Copilotille annettuun kommenttiin ja koodiin.

Kuva 14 Yksikkötestien generointi: Komennon ja ohjelmakoodin syöttö ChatGPT:lle


 Create 4 unit test cases for the calculator including arrange, act, and assert:

```
def addition(a, b):
    c = (a + b)
    return int (c)

def subtraction(a, b):
    c = (a - b)
    return int (c)

def multiplication(a, b):
    c = (a * b)
    return int (c)

def division(a, b):
    c = (a / b)
    return int (c)
```

 Sure, here are four unit test cases for the calculator functions:

Kuva 15 näyttää ChatGPT:n generoimat yksikkötestit laskinohjelmalle. Päällisin puolin vastaukset näyttävät oikeilta ja hyvin saman kaltaisilta kuin Copilotin luomat vastaukset. Mikäli yksikkötestit kopioidaan Visual Studio Codeen ja ajetaan läpi, voidaan kuitenkin todeta, että ohjelma palauttaa neljä virhettä. Tämä johtuu siitä, että yksikkötestien funktioista puuttuu parametri "result". ChatGPT generoi kaikki testitapaukset yksittäin ja lisäsi jokaiseen yksityiskohtaiset selitykset siitä, mitä koodissa tapahtuu ilman erillistä pyyntöä. Tämä helpottaa huomattavasti käyttäjää ymmärtämään, miten yksikkötestitapausten logiikka toimii. On kuitenkin huomioitava, että ChatGPT tuottamat vastaukset ovat vielä erittäin puutteellisia. Mikäli ohjelmaa haluaisi ylipäätään testata koodieditorissa onnistuneesti, olisi siihen tehtävä vielä suhteellisen paljon lisäyksiä sekä muutoksia.

Kuva 15 ChatGPT:n generoimat yksikkötestitapaukset

## 1. Test case for addition:

Arrange: Two numbers to be added, a = 5 and b = 7.

Act: Call the addition function with a = 5 and b = 7.

Assert: The result should be 12.

```
python Copy code  
  
def test_addition():  
    # Arrange  
    a = 5  
    b = 7  
    # Act  
    result = addition(a, b)  
    # Assert  
    assert result == 12
```

## 2. Test case for subtraction:

Arrange: Two numbers to be subtracted, a = 10 and b = 3.

Act: Call the subtraction function with a = 10 and b = 3.

Assert: The result should be 7.

```
python Copy code  
  
def test_subtraction():  
    # Arrange  
    a = 10  
    b = 3  
    # Act  
    result = subtraction(a, b)  
    # Assert  
    assert result == 7
```

Regenerate response



### 4.2.3 Kolmas testitapaus: Ongelmanratkaisu

Kolmalnnessa tehtävässä ChatGPT:lle syötettiin sama viallinen ohjelmakoodi kuin Copilotille kuvassa 16 näkyvällä tavalla. Myös ChatGPT:lle esitettiin kysymykset, mitä vikaa koodissa on ja kuinka sen voisi korjata.



Kuva 16 Virheellisen ohjelmakoodin syöttö ChatGPT:lle

LA What is wrong with this code?  
How can I fix it?

```
const cars=();  
car[0]="Volvo";  
car[1]="BMW";  
car[2]="Saab";  
car[120]="Ford";  
car[101]="Fiat";  
  
for (let i=0;i<cars.length;i+){  
  consolelog(cars[i]);  
}
```

ChatGPT onnistui suorittamaan tehtävän täydellisesti toisin kuin Copilot, joka on tähän mennessä onnistunut aiemmissä tehtävissä ChatGPT:tä paremmin. ChatGPT onnistui löytämään ja korjaamaan kaikki neljä ohjelmakoodissa esiintynyttä virhettä, kuten kuvassa 17 on havainnollistettu. Jo aiemmin mainittujen virheiden lisäksi ChatGPT havaitsi koodista myös väärän muuttujanimen "car", jolla yritetään määrittää arvoja cars-nimiselle joukolle. Tapa, jolla ChatGPT kertoo havaitsemistaan virheistä ja esittää korjausehdotuksia on erittäin selkeä sekä käyttäjäystävällinen. Jokainen virhe on selkeästi numeroilla eroteltu ja käyttäjälle on ymmärrettävästi kerrottu, miten ja miksi virheet tulisi korjata.

Kuva 17 ChatGPT:n löytämät virheet sekä ratkaisuehdotukset

There are a few issues with this code:

- The array `cars` is initialized incorrectly. You need to use square brackets `[]` instead of parentheses `()` to create an empty array:
 

```
javascript Copy code
const cars = [];
```
- The variable name used to access the array is incorrect. You defined the variable as `cars` but then tried to access it using `car`. Change `car` to `cars` in the lines where you're assigning values to the array:
 

```
python Copy code
cars[0] = "Volvo";
cars[1] = "BMW";
cars[2] = "Saab";
cars[120] = "Ford";
cars[101] = "Fiat";
```
- The for loop is not incrementing the index correctly. Change `i+` to `i++`:
 

```
css Copy code
for (let i=0; i<cars.length; i++){
```
- The method to print to the console is misspelled. Change `console.log` to `console.log`:
 

```
javascript Copy code
console.log(cars[i]);
```

### 4.3 Tulosten analysointi

Ensimmäisestä testitapauksesta Copilot ja ChatGPT suoriutuivat melko hyvin ottaen huomioon, että niille ei annettu tarkentavia ohjeita. Molemmat tekoälyohjelmat onnistuivat luomaan toimivan ohjelman. Mutta mikäli tehtävän onnistumista arvioitaisiin puhtaasti pelkän tehtävänannon perusteella, olisi Copilot siinä tapauksessa voittaja. Copilot onnistui täydellisesti luomaan juuri sellaisen ohjelman kuin siltä pyydettiin, lisäämättä tai jättämättä pois mitään ominaisuuksia. ChatGPT taas teki pelistä huomattavasti käyttäjäystävällisemmän, mutta epäonnistui pelin voittologiikan luomisessa.

On kuitenkin otettava huomioon, että molempien tekoälyohjelmien generoimat koodit olivat vain ensimmäiset versiot, jotka perustuivat puhtaasti tehtävänantoon. Molempien ohjelmien tuottamia vastauksia voisi helposti muokata haluttuun suuntaan antamalla lisäohjeita tai pilkkomalla tehtävää vielä pienempiin osiin. Myös sillä, kuinka tekoälyohjelmille annetut käskyt on muotoiltu, on erittäin suuri vaikutus niiden tuottamiin vastauksiin.

Koska Lucky7-tehtävä on suunniteltu ohjelmoinnin perusteet-kurssille, se on melko yksinkertainen varsinkin kokeneemmalle ohjelmoijalle. On kuitenkin hyvä huomioida, että vaikka tehtävän komennot on helppo syöttää Copilotille ja ChatGPT:lle, ohjelmoinnin perusteita suorittavan opiskelijan tulisi kuitenkin jo ymmärtää ohjelmoinnin peruskäsitteet hyvin, jotta hän voisi varmistua siitä, että generoidut koodit vastaavat tehtävänantoa. On myös tärkeää, että ohjelmoinnin opiskelija ymmärtäisi tekoälyohjelmien generoimien ohjelmistokoodien logiikan, jotta menetelmiä voisi hyödyntää myös tulevaisuudessa.

Toisesta testitapauksesta Copilot suoriutui huomattavasti paremmin kuin ChatGPT. Copilot suoriutui yksikkötestien luomisesta erittäin nopeasti ja tehtävänannon mukaisesti yhtä testiä lukuun ottamatta. Kun molempien tekoälyohjelmien generoimat ratkaisut ajettiin Visual Studio Codessa, Copilotin ratkaisu tuotti yhden virhetulostuksen, kun taas ChatGPT:n neljä virhetulostusta.

ChatGPT ei tässäkin tehtävässä ainoastaan yrittänyt ratkaista sille annettua tehtävää, vaan se myös lisäsi vastauksien yhteyteen selityksiä koodin termeistä sekä logiikasta ilman erillistä pyyntöä. Tämä auttaa erityisesti aloittelevia ohjelmoijia ymmärtämään tehtävän ratkaisun paremmin sekä oppimaan sen sijaan, että käyttäjä saisi pelkän vastauksen ymmärtämättä sen tarkoitusta. On kuitenkin huomioitava, että vaikka ChatGPT osasi luoda rungon ja selitykset yksikkötesteille, tulisi koodeja kuitenkin osata muokata vielä suhteellisen paljon, jotta ohjelman voisi ajaa läpi ilman virhetulostuksia. Mikäli ChatGPT:n tuottamasta vastauksesta haluaisi hyötyä, vaatisi se kriittistä suhtautumista tekoälyn tuottamia vastauksia kohtaa sekä ongelmanratkaisutaitoja.

Copilot ei lisännyt ChatGPT:n tavoin generoimaansa vastaukseen ylimääräisiä käyttäjää helpottavia selitteitä, mutta muuten suoriutui tehtävästä lähes täydellisesti, yhtä virhettä lukuun ottamatta. Kuten aiemmin mainittiin Copilotin generoima viimeinen yksikkötesti

”division” palautti virheen, sillä jakolaskun tulos ei ollut kokonaisluku. Copilot generoi kaikkiin yksikkötesteihin samat muuttujan arvot eli numerot 2 ja 3 kun taas ChatGPT käytti jokaisessa testissä eri lukuja. Todennäköisesti Copilot ei ottanut muuttujien arvojen generoimisessa huomioon sitä, että laskutoimitusten vastauksien tulisi olla kokonaislukuja vaan automaattisesti generoi kaikkiin yksikkötesteihin samat luvut yksikertaisuuden vuoksi. On kuitenkin mahdotonta sanoa, osasiko ChatGPT ottaa tämän huomioon vai oliko kyseessä vain hyvää tuuria.

Tähän mennessä Copilot on onnistunut huomattavasti paremmin vastausten generoimisessa kuin ChatGPT. Kuitenkin kolmannessa ja viimeisessä tehtävässä ChatGPT onnistui päihittämään Copilotin. ChatGPT onnistui löytämään virheellisestä ohjelmakoodista kaikki neljä virhettä, kun taas Copilot löysi ainoastaan kolme.

Copilotin tapa esittää virheet sekä korjausehdotukset VS codessa on melko sekava. Vastauksia ei ole jäsennetty selkeästi ja tekstiä on hankala lukea. ChatGPT taas jäsenteli vastaukset erittäin helppolukuisiksi ja havainnollisti, kuinka virheet tulisi korjata sekä sanallisessa että koodi muodossa.

Yhteenvetona kaikkien kolmen vertailutehtävän tuloksista voi päätellä, että tekoälyohjelmat eivät pysty korvaamaan ihmisen omaa ajatteluprosessia, sillä niiden generoimat vastaukset vaativat kriittistä suhtautumista ja ongelmanratkaisutaitoja, jotta niitä voitaisiin hyödyntää mahdollisimman tehokkaasti. Tämä johtopäätös on merkittävä, sillä se osoittaa, että vaikka tekoälyllä onkin paljon potentiaalia ohjelmoinnin maailmassa, sen rooli ei voi olla täysin korvaava ihmisen ajattelulle ja ratkaisutaidolle. Sen sijaan tekoälyllä voi olla tärkeä rooli opiskelijoiden oppimisprosessin tukemisessa ja se voi tarjota uusia näkökulmia sekä ideoita.

On myös tärkeää huomata, että tekoälyohjelmien kyvyt kehittyvät jatkuvasti, ja tulevaisuudessa ne voivat mahdollisesti tarjota vielä laajempia mahdollisuuksia opetuksessa. Kuitenkin on tärkeää jatkaa tutkimusta ja kehitystyötä sen suhteen, miten tekoälyä voidaan hyödyntää opetuksessa mahdollisimman tehokkaasti ja miten sen käyttöä voidaan parhaiten integroida opetussuunnitelmaan.

## 5 Copilotin ja ChatGPT: hyödyntäminen ohjelmoinnin opetuksessa

Tämä luku käsittelee sitä, miten Hämeenlinnan ammattikorkeakoulun tietojenkäsittelyn opettajat suhtautuvat uusiin tekoälyä hyödyntäviin sovelluksiin, Copilottiin ja ChatGPT:hen. Tutkimus suoritettiin teemahaastatteluina, joissa opettajilta kerättiin mielipiteitä liittyen tekoälyohjelmien hyödyntämiseen ohjelmoinnin opetuksessa. Tutkimuksessa haluttiin selvittää, tulisiko opetusmenetelmiä muokata tekoälyohjelmien yleistyessä, voiko tekoälyohjelmien hyödyntämiseen liittyä riskejä ja mitä mahdollisuuksia tekoälyohjelmat voivat tarjota. Opettajien nimiä ei julkisteta opinnäytetyössä vaan heihin viitataan termeillä Opettaja A ja Opettaja B.

Opettajilta kysyttiin, mitä potentiaalia he näkevät Copilotissa sekä ChatGPT:ssä. Opettajan A mielestä Copilot tai ChatGPT eivät niin sanotusti yksinään riitä ratkomaan ongelmia vaan toimivat opiskelijalle enemmänkin hyvinä apu -ja työvälineinä: ”Ei voida olettaa, että tekoälyohjelmat kykenisivät itsenäisellä ajattelulla ratkomaan tehtäviä, sillä taustalle tarvitaan myös opiskelijan omaa arviointikykyä sekä ajatustyötä.” Myös Opettaja B on sitä mieltä, että Copilot ja ChatGPT ovat ensisijaisesti työkaluja, jotka nopeuttavat ohjelmointia ja pystyvät suorittamaan rutiininomaisia puuduttuvia tehtäviä, jättäen tilaa uusien asioiden oppimiselle sekä luovuudelle.

Tutkimuksessa haluttiin myös selvittää, miten Copilottia ja ChatGPT:tä voitaisiin hyödyntää ohjelmoinnin opetuksessa osana opetussuunnitelmaa. Opettaja A kertoi, että tekoälyohjelmien soveltamista tietojenkäsittelyn opetussuunnitelmassa tulisi suunnitella porrastetusti vuosikurssi kerrallaan: ”On otettava huomioon, että opintojen edetessä opiskelijalta vaaditaan enemmän loogista päättelykykyä sekä kykyä soveltaa aikaisemmin opittua tietoa. Tämän vuoksi on erittäin tärkeää, että opiskelijoille kerrotaan varhaisessa vaiheessa tekoälyohjelmien tuomista mahdollisuuksista samalla varmistuen, että tietoperusta hallitaan hyvin, jotta ongelmilta voitaisiin välttyä opintojen edetessä. Kun opiskelija hallitsee ohjelmoinnin alkeet ja logiikan, voi tekoälyä hyödyntää hedelmällisemmin vaativammassa ohjelmointitehtävissä apuvälineenä, esimerkiksi vaikeampien toiminallisuuksien lisäämiseen omaan työhön.” Ja vaikka oppilas saisikin esimerkiksi ChatGPT:ltä oikean vastauksen ohjelmointitehtävään se ei Opettajan A mielestä ole millään tavalla huono asia, sillä oppilas on kuitenkin osannut ajatella tehtävän pienempinä osina ja

muotoilla kysymykset oikein. Opettaja A kertoi myös, että esimerkiksi koneoppimisen ja tekoälyn kursseilla tekoälyohjelmia voidaan hyödyntää monellakin tavalla. Opettaja B oli myös sitä mieltä, että tekoälyohjelmia hyödyntämistä tulisi soveltaa oppimistason mukaisesti. ”Esimerkiksi opintojen alkupuolella ChatGPT:llä pystyy generoimaan itselleen ohjelmointiharjoituksia, ChatGPT:tä voi pyytää kommentoimaan, onko koodi järkevästi tehty ja sitä voi hyödyntää myös teoriapuolen ymmärtämisessä, pyytämällä sitä selittämään itselle vaikeita termejä sekä käsitteitä.” Opettajan B näkee, että kun perusymmärrys ohjelmoinnista on hallussa, tekoälyohjelmia pystyy pidemmällä opinnoissa hyödyntämään rutiininomaisten tehtävien automatisointiin, jotta oppilaalle jää aikaa oppia enemmän uusi taitoja sekä kehittää omaa soveltamiskykyään.

Koska tekoälyohjelmat ovat aiheuttaneet paljon huolta korkeakouluissa, opettajilta kysyttiin, tuleeko heille mieleen joitakin riskejä tai ongelmia, joita näiden tekoälyohjelmien hyödyntäminen osana ohjelmoinnin opetusta saattaisi aiheuttaa. Opettajan B näkee, että tekoälyohjelmien hyödyntäminen etenkin opintojen alkutaipaleella saattaa aiheuttaa merkittäviä riskejä oppilaan oppimisen kannalta: ”Opintojen alussa ohjelmointitehtävät ovat hyvin yksinkertaisia, selkeitä ja rajattuja, jotka on sitten tosi helppo syöttää tekoälyn ratkaistavaksi ja onnistumisprosentti on melko suuri. On se syy tekoälyn hyödyntämiseen tällaisissa tehtävissä sitten kiire, ymmärryksen puute tai halu saada parempia arvosanoja niin kyllä tällaiset 24/7 tavoitettavissa olevat koodinratkaisijat voivat varmasti olla ainakin osalle opiskelijoista todella houkuttelevia.” Opettaja B toteaa myös, että ongelmia tuottaa se, että käytännössä oppilaan ei tarvitse ymmärtää ohjelmoinnista juuri mitään voidakseen ratkaista ensimmäisen vuoden tehtäviä melko pitkälle. Opettaja A toi lisäksi esille mahdollisia ongelmia, joita tekoälyohjelmien hyödyntäminen apuvälineinä osana kurssisisältöä saattaisi aiheuttaa. ”Copilot ei ole ilmainen sovellus ja vaikka ChatGPT on vielä käytettävissä täysin ilmaiseksi, tulevaisuudesta ei voi koskaan tietää. Koulu ei pysty takaamaan kaikkien maksullisten ohjelmien hyödyntämistä oppilaille ilmaiseksi ja siksi meidän opettajien tehtävänä onkin suunnitella kurssisisällöt niin, että opetus ei ole riippuvainen jostain tietystä ohjelmasta, joka saattaisi yht. äkkiä muuttuakin vaikka maksulliseksi. Lisäksi sieltä löytyy näitä tekijänoikeusongelmia, jotka voivat joku päivää johtaa siihen, että nämä tekoälyohjelmat ajetaan alas, jonka vuoksi sidonnaisuutta tällaisiin ohjelmiin ei voida vielä opetussuunnitelman kannalta tehdä.”

Opettajia pyydettiin myös kertomaan, kuinka näitä ongelmia sekä riskejä pystyttäisiin minimoimaan tai ratkaisemaan. Molemmat opettajat olivat sitä mieltä, että asioita pyritään varmasti miettimään mutta oikeita vastauksia tuskin on olemassa. Haastateltavat opettajat jakavat saman mielipiteen siitä, että tulevaisuudessa opiskelijoille tulisi entistä enemmän painottaa perusymmärryksen sekä eettisen opiskelun tärkeyttä. Opettaja B kertoi, että yksi tapa varmistaa oppilaiden ymmärryksen tasoa, on itsenäisten tehtävien läpikäyminen suullisesti oppilaan kanssa: ”Arviointimenetelmänä keskustelu varmaan kasvaa entisestään, jolloin asioita pitää oikeasti selittää tehtävien tekemisen lisäksi”. Opettajan A mukaan pidemmälle opintoihin mentäessä kurssit on jo suunniteltu niin, että kurssiarvosana harvoin perustuu yksittäisten tehtävien varaan, sen sijaan oppilaan on osoitettava soveltamiskykyä esimerkiksi erilaisissa projekteissa: ” Jos puhutaan yksittäisistä tehtävistä niin sitä ennakointia tehdään jo nyt. Meilläkin on monesti projektitöitä ja muuta sovellettua kurssimateriaalia, missä sitä oppimista ei mitata yksittäisten tehtävien avulla. Ja sitten jos onkin sellainen kurssi, että siinä arviointi perustuu tenttiin, missä on kysymyksiä niin ne pitää sitten suunnitella silleen, että vaikka niissä saisikin tekoälyä käyttää sitä pitää osata käyttää työkaluna.”

Yhteenvetona teemahaastattelututkimuksesta voidaan todeta, että vaikka kaikkien uusien työkalujen mukana tulee aina joitakin riskejä ja ongelmia, niitä pyritään ratkaisemaan ja pohtimaan jo aikaisessa vaiheessa. Opettajilla on suuri merkitys opetuksen laadullisessa toteuttamisessa sekä kurssisisältöjen suunnittelussa ja näiden lisäksi opettajien on oltava perillä kehittyvistä teknologioista, jotta opetussuunnitelmaa voidaan muokata niiden mukaisesti.

## 6 Johtopäätökset ja pohdinta

Opinnäytetyön aihe on erittäin mielenkiintoinen ja ajankohtainen, sillä tekoälyteknologian kehitys kiihtyy ja vaikuttaa monilla eri tavoilla yhteiskuntaan. Tämän tutkimuksen avulla voidaan paremmin ymmärtää tekoälyn vaikutuksia ja mahdollisuuksia sen hyödyntämiseen ohjelmoinnin opetuksessa.

Opinnäytetyössä vertailtiin kahta eri tekoälytyökalua, Copilottia ja ChatGPT:tä sekä niiden vaikutusta ohjelmoinnin opetukseen. Työkalujen vertailu antoi arvokasta tietoa siitä, miten erilaiset tekoälysovellukset voivat auttaa ohjelmoinnin oppimisessa ja kuinka niiden erilaisia ominaisuuksia voidaan hyödyntää tehokkaasti. Vertailun tulokset voivat auttaa opettajia ja oppilaitoksia valitsemaan sopivimman tekoälytyökalun ohjelmoinnin opetuksen tueksi.

Teemahaastattelumenetelmä mahdollisti opettajien näkemysten ja kokemusten keräämisen tekoälytyökalujen käytöstä ohjelmoinnin opetuksessa. Haastattelujen avulla saatiin arvokasta tietoa siitä, miten tekoälytyökaluja voidaan hyödyntää opetuksessa ja millaisia haasteita niiden käyttöön liittyy.

Vertailun ja teemahaastattelujen tulokset yhdessä ovat hyödyllisiä ohjelmoinnin opetuksen suunnittelussa, kun pohditaan millaisia tehtäviä ja haasteita opiskelijoille annetaan. Opetuksen kannalta on ensiarvoisen tärkeää suunnitella ohjelmoinnin kurssisisältöjä niin, että ne vastaavat opiskelijoiden ohjelmointitaitojen tasoa ja tarjoavat haasteita. Tämä auttaa varmistamaan, että opiskelijat voivat edetä opintojensa kanssa sopivassa tahdissa ja oppia uusia taitoja tehokkaasti.

Lisäksi on tärkeää valita oikeanlainen tekoälytyökalu tukemaan opiskelijoiden oppimisprosessia. Tämä voi auttaa opiskelijoita ymmärtämään abstrakteja käsitteitä, ratkaisemaan ongelmia ja kehittämään luovaa ajattelua. Tekoälytyökalut voivat myös tarjota arvokkaita opetusresursseja, kuten automaattisia arviointijärjestelmiä ja oppimispeljä, jotka voivat auttaa opiskelijoita kehittämään taitojaan hausalla ja motivoivalla tavalla.



## 6.1 Vertailututkimus

Opinnäytetyön tietoperustassa käsiteltiin laajasti tekoälyn keskeisiä käsitteitä sekä tarkemmin kahta koneoppimista hyödyntävää ohjelmaa: Copilotia ja ChatGPT:tä. Tämä antoi vankan pohjan näiden tekoälyohjelmien vertailulle. Ennen vertailututkimuksen toteuttamista oli jo selvillä, miten nämä ohjelmat toimivat, millaisia ominaisuuksia niillä on, ja kuinka ne eroavat toisistaan ohjelmakoodin generoimisen, käyttöliittymän ja komentojen syötön suhteen. Vertailututkimuksen ja tietoperustan avulla saatiin kattava vastaus tutkimuskysymykseen ”Miten Copilot ja ChatGPT eroavat toisistaan?”. Vertailututkimuksen tulokset vahvistavat tietoperustassa käsitellyjä asioita ja tämän perusteella voidaan todeta, että vertailututkimus on onnistunut ja paikkaansa pitävä.

Copilotin ja ChatGPT:n vertailun tuloksista voi päätellä, että Copilot sopii huomattavasti paremmin ohjelmointiin, mikäli käyttäjällä on jo kokemusta ohjelmoinnin perustermeistä sekä taitoa soveltaa yksinkertaisia ohjelmointirakenteita vaativammassa tehtävissä sekä kokonaisuuksissa. Copilottia voi hyödyntää erittäin tehokkaasti ohjelmointityökaluna, kun sille on annettu selkeä konteksti, joka voi olla esimerkiksi tarkka kuvaus ohjelmakoodista tai sen sisältämistä toiminnallisuuksista eli funktioista. Tämän takia Copilot soveltuu huomattavasti ChatGPT:tä paremmin hyödynnettäväksi osana ohjelmoinnin opetusta pidemmällä opinnossa, kun oppilas hallitsee ohjelmoinnin teorian ja on valmis siirtymään haastavampiin ja suurempiin ohjelmistokehityksen kokonaisuuksiin.

ChatGPT:n vahvuus on tuottaa luovia ja monipuolisia ratkaisuehdotuksia, mutta sen kyky soveltaa ohjelmointirakenteita on rajallisempi. ChatGPT:tä voidaan hyödyntää erityisesti luovan ideointiprosessin tukena, termien selittämisessä ja esimerkiksi ongelmanratkaisutehtävien yhteydessä, mutta sen käyttö ja vastauksien tulkitseminen on haastavampaa ohjelmointitehtävissä. ChatGPT:n hyödyntäminen sopii erityisesti ohjelmoinnin opetuksen alkuvaiheeseen, esimerkiksi ensimmäisellä vuosikurssilla, kun opiskelijoilla ei välttämättä ole vielä laajaa kokemusta ohjelmoinnin peruskäsitteistä tai ohjelmointirakenteista. Tämä voi kuitenkin aiheuttaa merkittäviä riskejä oppimisen kannalta, mikäli ChatGPT:tä ei hyödynnetä eettisesti opintojen ohella.

Syy siihen, miksi ChatGPT:n generoimat vastaukset ovat selkeämpiä ja helpommin ymmärrettäviä, piilee tekoälyohjelmien opetusdatassa. Koska Copilotin opetusdata on suurimmaksi osaksi avointa lähdekoodia, sillä ei ole samanlaisia resursseja tuottaa selkeitä lauserakenteita tai ylipäätään tavalliseen tekstiin pohjautuvia vastauksia kuin ChatGPT:llä. ChatGPT on varta vasten koulutettu vastaamaan käyttäjän kysymyksiin selkeästi sekä ihmistä muistuttavalla tavalla, toisin kuin Copilot, joka on koulutettu lähtökohtaisesti generoimaan ja tulkitsemaan ohjelmakoodia.

Lisäksi on huomattava, että Copilotin käyttöliittymä on hyvin erilainen verrattuna ChatGPT:n käyttöliittymään. Copilot hyödyntää IDE-pohjaisia käyttöliittymiä, jotka ovat ohjelmointityökalujen standardeja. Tämä tarkoittaa, että Copilotin käyttö vaatii käyttäjältä jo perusosaamista ohjelmoinnin perusasioista ja erilaisista ohjelmointirakenteista, jotta käyttäjä pystyy ymmärtämään Copilotin generoimaa koodia ja soveltamaan sitä omassa ohjelmointityössään. Toisaalta ChatGPT:n käyttöliittymä on käyttäjäystävällisempi ja helpompi ymmärtää myös ohjelmointitaidottomille käyttäjille, sillä se käyttää tavallista tekstipohjaista käyttöliittymää.

## 6.2 Teemahaastattelututkimus

Tietoperustassa esitettiin myös erilaisista lähteistä kerättyjä, opettajien ja oppilaiden mielipiteitä tekoälyohjelmien vaikutuksesta oppimiseen ja opettamiseen tähän mennessä. Tietoperustassa esiin tulleet mielipiteet antoivat suuntaa teemahaastattelujen kysymysten muotoilemiselle sekä sisällön rajaamiselle. Myös tekoälyohjelmien eettisiä ongelmia käsiteltiin teemahaastattelun lisäksi jo teoriaosuudessa. Haastattelututkimus vastasi käytännönläheisemmin ja kattavammin tutkimuskysymyksiin ”Kuinka tekoälyohjelmia voitaisiin hyödyntää ohjelmoinnin opetuksessa?” ja ”Minkälaisia riskejä tekoälyn hyödyntämiseen osana ohjelmoinnin opetusta liittyy?” kuin teoriaosuudessa esitetyt vastaukset.

Teemahaastatteluissa korostui kahden tietojenkäsittelyn opettajan mielipide siitä, kuinka tekoälyohjelmiin tulisi suhtautua lähtökohtaisesti uusina työkaluina. Opettajien mielipiteet myös vahvistivat vertailun tuloksia siitä, miten tekoälyohjelmia voidaan hyödyntää monilla eri tavoilla porrastetusti osana tietojenkäsittelyn koulutuksen opetussuunnitelmaa.

Opettajat painottivat myös, että tekoälyohjelmien käytön tulisi olla eettisesti kestävää ja ohjeistuksen mukaista, jotta opiskelijoiden oppimisprosessi ei kärsi.

On tärkeää ymmärtää, että tekoälyohjelmat voivat olla arvokkaita oppimisen työkaluja, eikä niitä tulisi kieltää oppilaitoksissa muutosvastaisuuden tai uhkakuvien takia. Kieltämisen sijaan, tekoälyohjelmien tuomista hyödyistä sekä haitoista tulisi keskustella avoimesti opiskelijoiden kanssa, jotta niistä voisi saada maksimaalista hyötyä oppimisen avuksi. Tekoälyohjelmat voivat tarjota ohjelmoinnin opiskelijoille mahdollisuuden harjoitella koodaamista ja ongelmanratkaisutaitoja lukuisilla eri tavoilla. Parhaimmassa tapauksessa tekoälyohjelmat voivat antaa tukea ohjelmoinnin oppimiseen luokkatoverin tai opettajan tavoin ja niiden hyödyntämistä voidaan räätälöidä yksilöllisiin oppimistavoitteisiin sopivaksi.

Tutkimus osoitti, että tulevaisuudessa oppilaitoksissa tulisi entistä enemmän korostaa eettisen opiskelun ja lähdekriittisyyden merkitystä opetussuunnitelmassa. Tämä on erityisen tärkeää tekoälyohjelmien käytön yleistyessä. Opiskelijoille tulisi tarjota ohjeistusta siitä, kuinka tekoälyohjelmia voidaan käyttää eettisesti ja miten niiden generoimia vastauksia voidaan tulkita kriittisesti. On tärkeää painottaa, että tekoälyohjelmia ei voida käyttää kiertotienä oppimiseen, vaan ihmisen omaa ajattelukykyä tarvitaan edelleen niiden hyödyntämisessä. Tämä tulee huomioida myös oppilaitosten ohjelmoinnin opetuksessa ja käytännön harjoituksissa.

## 7 Yhteenveto

Kaikkiin tutkimuskysymyksiin on tämän opinnäytetyön aikana saatu vastattua kattavasti sekä opinnäytetyön teoriaosuudessa että vertailu- ja teemahaastattelututkimuksen yhteydessä. Opinnäytetyön teoriaosuus vastaa jo osittain kaikkiin kolmeen tutkimuskysymykseen ja toiminnallisessa osuudessa vastauksiin on tuotu laajasti uusia näkökulmia sekä pohdintaa. Opinnäytetyön teoriapohja koostuu todella monesta erilaisesta lähteestä ja koska tutkimustulokset ovat tukevat opinnäytetyön tietoperustan sisältöä, voidaan vastausten todeta olevan melko luotettavia.

Tämän tutkimuksen perusteella voidaan todeta, että Copilotin ja ChatGPT:n erot vaikuttavat ratkaisevasti siihen, minkälaisissa tehtävissä niiden hyödyntäminen on kaikista tuloksellisinta sekä järkevintä. Tutkimus osoittaa myös, että tekoälyohjelmien yleistyessä ja kehittyessä opetussuunnitelmaa sekä opetusmetodeja tulisi muokata niin, että tekoälytyökalujen hyödyntäminen osana ohjelmoinnin opetusta ei vaikuttaisi negatiivisesti oppimiskehitykseen.

Opinnäytetyöprosessin aikana opin paljon uusia asioita. Koska halusin koota opinnäytetyön tietoperustaan laajan kuvauksen tekoälystä yleisesti sekä trendaavista Copilotista ja ChatGPT:stä, sain pintaa syvällisemmän käsityksen teknologioista ja menetelmistä tekoälyohjelmien takana. Opin myös hyödyntämään Copilottia ja ChatGPT:tä tehokkaasti sekä eettisesti ja sain opettajilta hyödyllistä informaatiota tekoälyohjelmien hyödyistä ja haitoista opetuksen näkökulmasta.

Koska tekoälytyökalujen hyödyntäminen oppilaitoksissa on edelleen verrattain uusi ilmiö, tulevaisuudessa olisi kiinnostavaa tutkia niiden vaikutuksia oppimiseen perusteellisemmin ja konkreettisemmin. Erityisesti olisi mielenkiintoista selvittää, millaisia vaikutuksia tekoälytyökaluilla on opiskelijoiden kykyyn sisäistää ohjelmoinnin teoriaa ja ratkaista ongelmia sekä tutkia, lisääkö tekoälytyökalujen hyödyntäminen opiskelun mielekkyyttä sekä luovuutta.

## Lähteet

- Addagatla, A. (2021, huhtikuuta 20). A Study of Artificial Neural Networks (ANN). *The Startup*. <https://medium.com/swlh/a-study-of-artificial-neural-networks-ann-7e1f7f2ac891>
- Agrawal, A., Gans, J. S., & Goldfarb, A. (2017). What to Expect From Artificial Intelligence. *MIT SLOAN MANAGEMENT REVIEW*.
- Ailisto, H., Heikkilä, E., Helaakoski, H., Neuvonen, A., & Seppälä, T. (2018, kesäkuuta 19). *Tekoälyn kokonaiskuva ja osaamiskartoitus* [Raportti]. Valtioneuvoston kanslia. <https://julkaisut.valtioneuvosto.fi/handle/10024/160925>
- Burns, E., & Brush, K. (2021). *What is Deep Learning and How Does It Work?* Enterprise AI. <https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network>
- Castrounis, A. (2022). *AI Explained*. Why of AI. <https://www.whyofai.com/blog/ai-explained>
- Copeland, M. (2016, heinäkuuta 29). *The Difference Between AI, Machine Learning, and Deep Learning?* NVIDIA Blog. <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>
- Cotton, D. R. E., Cotton, P. A., & Shipway, J. R. (2023). *Chatting and Cheating. Ensuring academic integrity in the era of ChatGPT*. EdArXiv. <https://doi.org/10.35542/osf.io/mrz8h>
- Council of Europe. (2023). *History of Artificial Intelligence—Artificial Intelligence—Publi.coe.int*. Artificial Intelligence. <https://www.coe.int/en/web/artificial-intelligence/history-of-ai>
- Dakhel, A. M., Majdinasab, V., Nikanjam, A., Khomh, F., Desmarais, M. C., Ming, Z., & Jiang. (2022). *GitHub Copilot AI pair programmer: Asset or Liability?* (arXiv:2206.15331). arXiv. <https://doi.org/10.48550/arXiv.2206.15331>

- Devaney, P. (2022, joulukuuta 30). *What is ChatGPT and how to use the AI tool effectively—GHacks Tech News*. GHacks Technology News.  
<https://www.ghacks.net/2022/12/30/what-is-chatgpt-how-to-use-chatgpt/>
- Fitzpatrick, S. (2022). *On the Nature of AI Code Copilots*.
- Friedman, N. (2021, kesäkuuta 29). Introducing GitHub Copilot: Your AI pair programmer. *The GitHub Blog*. <https://github.blog/2021-06-29-introducing-github-copilot-ai-pair-programmer/>
- GitHub. (2022). *GitHub Copilot · Your AI pair programmer*. GitHub.  
<https://github.com/features/copilot>
- GitHub. (2023). *About GitHub Copilot*. GitHub Docs. [https://ghdocs-prod.azurewebsites.net/\\_next/data/S\\_iTghc4zogItl6u1ly8k/en/free-pro-team@latest/copilot/overview-of-github-copilot/about-github-copilot.json?versionId=free-pro-team%40latest&productId=copilot&restPage=overview-of-github-copilot&restPage=about-github-copilot](https://ghdocs-prod.azurewebsites.net/_next/data/S_iTghc4zogItl6u1ly8k/en/free-pro-team@latest/copilot/overview-of-github-copilot/about-github-copilot.json?versionId=free-pro-team%40latest&productId=copilot&restPage=overview-of-github-copilot&restPage=about-github-copilot)
- Grincalaitis, M. (2022, joulukuuta 12). ChatGPT vs GPT-3 by Merun. *Medium*.  
<https://merunasgrincalaitis.medium.com/chatgpt-vs-gpt-3-by-merun-4c282c83d50a>
- Hughes, A. (2023). *ChatGPT: Everything you need to know about OpenAI's GPT-3 tool*. BBC Science Focus Magazine. <https://www.sciencefocus.com/future-technology/gpt-3/>
- Kalliamvakou, E. (2022, syyskuuta 7). Research: Quantifying GitHub Copilot's impact on developer productivity and happiness. *The GitHub Blog*. <https://github.blog/2022-09-07-research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/>

- Lampinen, A. (2019). *Algoritmit suodattavat ja suosittelevat*. Mannerheimin Lastensuojeluliitto. <https://www.mll.fi/vanhemmille/tietoa-lapsiperheen-elamasta/hyvinvointia-digiajassa/algoritmit-suodattavat-ja-suosittelevat/>
- Li, H. Y., An, J. T., & Zhang, Y. (2021). Ethical Problems and Countermeasures of Artificial Intelligence Technology. *E3S Web of Conferences*, 251, 01063. <https://doi.org/10.1051/e3sconf/202125101063>
- Lønvik, A. T. (2021). *An overview of Artificial Intelligence; Application, user experience and ethical problems*.
- OpenAI. (2015, joulukuuta 11). *About OpenAI*. OpenAI. <https://openai.com/about/>
- OpenAI. (2022, marraskuuta 30). *ChatGPT: Optimizing Language Models for Dialogue*. OpenAI. <https://openai.com/blog/chatgpt/>
- Parmar, A. (2022, toukokuuta 23). How Artificial Intelligence Improves Software Development. *RTInsights*. <https://www.rtinsights.com/how-artificial-intelligence-improves-software-development/>
- Pesonen, R. (2021, elokuuta 25). *Tekoäly*. filosofia.fi. <https://filosofia.fi/fi/ensyklopedia/tekoaly>
- Pietikäinen, M. (2021). *Tekoälyn haasteet: Koneoppimisesta ja konenäöstä tunnetekoälyyn*.
- Pitkänen, L., & Leponiemi, T. (2023, tammikuuta 16). *Googlea taitavampi tekoäly voi ajaa oppilaitokset pulaan – ”Viidessä minuutissa minulla oli kiitettävä vastaus”, sanoo yllättynyt opettaja*. Yle Uutiset. <https://yle.fi/a/74-20011922>
- Pocock, K. (2023, tammikuuta 23). *What Is Chat GPT? PC Guide*. <https://www.pcguides.com/apps/what-is-chat-gpt/>
- Popenici, S. A. D., & Kerr, S. (2017). Exploring the impact of artificial intelligence on teaching and learning in higher education. *Research and Practice in Technology Enhanced Learning*, 12(1), 22. <https://doi.org/10.1186/s41039-017-0062-8>

Prokopets, M. (2021, marraskuuta 14). *The Ultimate Manual to GitHub Copilot*. Nira.

<https://nira.com/github-copilot/>

Rytkönen, E. (2023, tammikuuta 17). ”Steroideilla buustattua hakukonetta” hyödynnetään jo korkeakoulun opetuksessa. Yle Uutiset. <https://yle.fi/a/74-20012508>

Saçan. (2023, tammikuuta 24). *ChatGPT shakes up education: “Time for meaningful dialogue”* | SURF Communities. [https://communities.surf.nl/en/ai-in-](https://communities.surf.nl/en/ai-in-education/article/chatgpt-shakes-up-education-time-for-meaningful-dialogue)

[education/article/chatgpt-shakes-up-education-time-for-meaningful-dialogue](https://communities.surf.nl/en/ai-in-education/article/chatgpt-shakes-up-education-time-for-meaningful-dialogue)

SITNFlash. (2017, elokuuta 28). The History of Artificial Intelligence. *Science in the News*.

<https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/>

Sojonen, E. (2021, maaliskuuta 31). Mitä koronan jälkeen – korkeakoulujen alettava

suunnitella toimintaa koronan jälkeen. *Akava*. [https://akava.fi/blogit/mita-koronan-](https://akava.fi/blogit/mita-koronan-jalkeen-korkeakoulujen-alettava-suunnitella-toimintaa-koronan-jalkeen/)

[jalkeen-korkeakoulujen-alettava-suunnitella-toimintaa-koronan-jalkeen/](https://akava.fi/blogit/mita-koronan-jalkeen-korkeakoulujen-alettava-suunnitella-toimintaa-koronan-jalkeen/)

Somoye, F. L. (2023, tammikuuta 27). *Why is Chat GPT at capacity?* PC Guide.

<https://www.pcguides.com/apps/chat-gpt-at-capacity/>

St. George, B. (2021). *What is the Turing Test?* Enterprise AI.

<https://www.techtarget.com/searchenterpriseai/definition/Turing-test>

Susnjak, T. (2022). *ChatGPT: The End of Online Exam Integrity?* (arXiv:2212.09292). arXiv.

<http://arxiv.org/abs/2212.09292>

Thompson, T. (2021). *Turing Test*. <https://www.theologyai.com/2021/09/turing-test.html>

Työ- ja elinkeinoministeriön tekoälyohjelman ohjausryhmä. (2017, lokakuuta 23). *Suomen tekoälyaika: Suomi tekoälyn soveltamisen kärkimaaksi: Tavoite ja*

*toimenpidesuosituks* [TEM raportteja]. Työ- ja elinkeinoministeriö.

<https://julkaisut.valtioneuvosto.fi/handle/10024/80849>



Ammattikorkeakoulujen rehtorineuvosto Arene ry. (2020). *Ammattikorkeakoulujen opinnäytetyöiden eettiset suositukset*. Noudettu osoitteesta [http://www.arene.fi/wp-content/uploads/Raportit/2020/AMMATTIKORKEAKOULUJEN%20OPINN%C3%84YTE%20T%C3%96IDEN%20EETTISET%20SUOSITUKSET%202020.pdf?\\_t=1578480382](http://www.arene.fi/wp-content/uploads/Raportit/2020/AMMATTIKORKEAKOULUJEN%20OPINN%C3%84YTE%20T%C3%96IDEN%20EETTISET%20SUOSITUKSET%202020.pdf?_t=1578480382)

Tutkimuseettinen neuvottelukunta. (2012). *Hyvä tieteellinen käytäntö ja sen loukkausepäilyjen käsitteleminen Suomessa*. Noudettu osoitteesta <https://tenk.fi/fi/ohjeet-ja-aineistot/HTK-ohje-2012>

Tutkimuseettinen neuvottelukunta. (2019). *Ihmiseen kohdistuvan tutkimuksen eettiset periaatteet ja ihmistieteiden eettinen ennakoarviointi Suomessa*. Noudettu osoitteesta [https://tenk.fi/sites/tenk.fi/files/Ihmistieteiden\\_eettisen\\_ennakoarvioinnin\\_ohje\\_2019.pdf](https://tenk.fi/sites/tenk.fi/files/Ihmistieteiden_eettisen_ennakoarvioinnin_ohje_2019.pdf)

**Liite 1: Aineistonhallintasuunnitelma**

Tutkimusaineistoa kerättiin Copilotin ja ChatGPT:n vertailututkimuksessa sekä teemahaastatteluissa, joissa kahdelta Hämeen ammattikorkeakoulun tietojenkäsittelyn koulutuksen opettajalta kerättiin mielipiteitä tekoälyohjelmien soveltamiseen opetuksessa liittyvistä mahdollisuuksista ja riskeistä.

Vertailututkimuksesta saatua aineistoa havainnollistetaan opinnäytetyössä kuvankaappauksilla sekä otteilla ohjelmakoodista, jotka säilytetään tekijän tietokoneen C-asemalla. Näistä tehdään säännöllisesti varmuuskopiot opinnäytetyön tekijän ulkoiselle kiintolevyllä.

Tutkimushaastattelu oli opettajille vapaaehtoinen ja tutkimuksen tuloksissa käsiteltäviä henkilötietoja olivat paikkakunta sekä nimike. Haastattelut pidettiin videopuhelun välityksellä ja ne nauhoitettiin, litteroitiin docx-tiedostoihin sekä tallennettiin tekijän paikalliselle C-asemalle. Näistä tehdään säännöllisesti varmuuskopiot opinnäytetyön tekijän ulkoiselle kiintolevyllä.

Vertailututkimuksesta saatua aineistoa sekä haastattelutiedostoja säilytetään 1 vuosi opinnäytetyön valmistumisesta, jonka jälkeen ne hävitetään. Opinnäytetyön tekijä omistaa tutkimusaineistot sekä tulokset.

## **Liite 2: Teemahaastattelujen pääkysymykset**

1. Mitä potentiaalia näet tekoälyä hyödyntävissä Copilotissa ja ChatGPT:ssä yleisesti ohjelmointia ajatellen?
2. Miten Copilottia ja ChatGPT:tä voitaisiin hyödyntää ohjelmoinnin opetuksessa osana opetussuunnitelmaa?
3. Tuleeko mieleesi joitakin riskejä tai ongelmia, joita näiden kyseisten tekoälyohjelmien hyödyntäminen osana ohjelmoinnin opetusta saattaisi aiheuttaa?
4. Miten näitä riskejä sekä ongelmia pystyttäisiin minimoimaan ja ratkaisemaan?