Tampereen yliopisto

Maisa Latva

LUKION VALINNAISEN MATEMATIIKAN KURSSIN TUKIMATERIAALIN KEHITTÄMINEN HYÖDYNTÄEN OHJELMOINTIA

Pro gradu -tutkielma
Informaatioteknologian ja viestinnän tiedekunta
Huhtikuu 2024

TIIVISTELMÄ

Maisa Latva: Lukion valinnaisen matematiikan kurssin tukimateriaalin kehittäminen hyödyntäen ohjelmointia
Pro gradu -tutkielma
Tampereen yliopisto
Matematiikan maisteriohjelma
Huhtikuu 2024

Tässä Pro gradu -tutkielmassa kehitettiin tukimateriaali lukion pitkän matematiikan opintojaksolle, MAA12 – Analyysi ja jatkuva jakauma, hyödyntäen Python-ohjelmointia. Lukion opetussuunnitelman perusteiden 2019 mukaan lukioiden tulee tarjota valinnainen pitkän matematiikan opintojakso MAA11–Algoritmit ja lukuteoria, johon kuuluu merkittävänä osana ohjelmointi ja ohjelmoinnillinen ajattelu. Tutkielmassa tuotettu materiaali rakentuu jo opitun tiedon päälle, mutta tarjoaa uusia tapoja hyödyntää ohjelmointia sekä kehittää ohjelmoinnillista ajattelua ja ongelmanratkaisutaitoja. Esitietoina oletetaan olevan edellä mainittu MAA11-opintojakso tai sitä vastaavat pohjatiedot.

Materiaali sisältää teoria- ja tehtäväpaketin, jonka avulla ohjelmointi voidaan yhdistää MAA12-opintojakson aiheisiin. Tämän lisäksi materiaaliin kuuluu alku- ja loppukysely. Ennen opintojaksoa tehtävällä alkukyselyllä kartoitetaan opiskelijoiden mielenkiintoa ohjelmointia kohtaan sekä pohjatietoja aiheesta. Alkukysely voidaan toteuttaa opintojakson ensimmäisillä opetuskerroilla eikä se vaadi opiskelijoilta erillistä valmistautumista. Loppukyselyn vastauksien avulla voidaan tarkastella tukimateriaalin teoriaosuuksien selkeyttä ja tehtävien vaikeusastetta. Vastausten perusteella oppimateriaalipakettia on mahdollista kehittää seuraavia opetuskertoja varten.

Teoria- ja tehtäväpaketti on laadittu siten, että se sisältää tehtäviä koskien derivointia, integrointia, raja-arvoja sekä jatkuvasti jakautuneen satunnaismuuttujan odotusarvoa ja keskihajontaa. Tehtävätyypit on valittu tarjoamaan mahdollisimman monipuolisia tapoja kehittää ohjelmoinnillista ja matemaattista ajattelua, mutta samalla kuitenkin muodostaen yhtenäisen kokonaisuuden. Materiaalin teoriaosuus sisältää pääsääntöisesti ohjeita koodin kirjoittamiseen ja taustatietoa tehtävissä tarvittavasta logiikasta.

Tutkielman matematiikkaosuus käänteisfunktiosta on valittu vastaamaan yhtä MAA12-opintojakson oppimistavoitetta.

Avainsanat: matematiikka, raja-arvo, integraali, käänteisfunktio, tietotekniikka, ohjelmointi, Python-ohjelmointi, ohjelmoinnillinen ajattelu

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

LOPS2016	Lukion opetussuunnitelman perusteet 2016
LOPS2019	Lukion opetussuunnitelman perusteet 2019
MAA11	Lukion pitkän matematiikan opintojakso: Algoritmit ja lukuteoria
MAA12	Lukion pitkän matematiikan opintojakso: Analyysi ja jatkuva jakauma
$\mathbb{N}$	Luonnolliset luvut
Python	Ohjelmointikieli
$\mathbb{R}$	Reaaliluvut

1. JOHDANTO

Sana *ohjelmointi* herättää helposti mielikuvan tietokoneella istuvasta henkilöstä, jonka ruudulla juoksevat pitkät tekstirimpsut. Ohjelmointia on kuitenkin monenlaista ja jokainen meistä kohtaa sitä elämässään tavalla tai toisella. Yksinkertaisimmillaan ohjelmointi on ohjeiden noudattamista, reseptin lukemista tai rakennuspalikoilla rakentamista. Ohjelmoinnin opettamisen tarkoituksena ei ole tuottaa vain tehokkaita ohjelmakoodin kirjoittajia, vaan kehittää jokapäiväisessä elämässä tarvittavia taitoja.

Lukion opetussuunnitelman perusteiden 2019 nojalla koulujen tulee tarjota valinnaista pitkän matematiikan opintojaksoa MAA11–Algoritmit ja lukuteoria, jonka yhtenä oppimistavoitteena on yksinkertaisen algoritmin ohjelmointi [19, s. 229]. Tällainen tehtävä on helppo yhdistää matematiikan opiskeluun, sillä matematiikka tarjoaa lukemattomia vaihtoehtoja yksinkertaisista algoritmeista. Esimerkiksi voidaan valita logiikan konnektiivit ja luoda niitä vastaava ehtolauserakenne. Ohjelmointiopetukseen liittyviksi haasteiksi on esitetty esimerkiksi valmiiden materiaalien puuttumisen sekä ajan puutteen matematiikan opetuksen ohessa [25, s. 81].

Tämän Pro gradu -tutkielman tavoitteena on luoda Python-ohjelmointia sisältävä tukimateriaali toiselle lukion valinnaiselle pitkän matematiikan opintojaksolle ja näin jatkaa MAA11-opintojakson oppeja ohjelmoinnillisesta ajattelusta. Samalla materiaali antaa uusia tapoja käyttää ohjelmointia osana matematiikan opetusta. Tukimateriaalin tehtävät on rakennettu siten, että ne tukevat kurssin matemaattisia aiheita ja voivat toimia osana viikkotehtäviä.

Tutkielman teoreettinen viitekehys rakentuu matematiikan oppimisen, ohjelmoinnillisen ajattelun ja ohjelmoinnin oppimisen ympärille. Näistä asioista ja niiden linkittymisestä yhteen kerrotaan luvuissa 2 ja 3. Tutkielman matemaattinen osuus käsittelee käänteisfunktiota ja sen derivaattaa ja tämä on esitetty luvussa 4. Luvussa 5 esitetään perustelut tukimateriaalin teorialle ja tehtäville, ja miten niiden on tarkoitus tukea oppimista. Viimeisessä luvussa 6 pohditaan vielä, miten ohjelmoinnin opetusta voitaisiin jatkaa matematiikan parissa ja millaisia asioita tämän puitteissa tulee ottaa huomioon.

Tutkielman liitteenä on kehitetty tukimateriaali teoria- ja tehtäväosioineen. Sen lisäksi tutkielman liitteenä on opiskelijoiden pohjatietoja ja motivaatiota mittaava alkukysely sekä loppukysely, joka mittaa opittua ja antaa opettajalle työkaluja tukipaketin kehittämiseen.

2. MATEMATIIKAN OSAAMISEN LINKITTYMINEN OHJELMOINTIIN

Tässä luvussa perehdytään matematiikan osaamiseen ja sen yhteyteen ohjelmoinnilliseen ajatteluun ja ohjelmointiin. Luvussa esitellään tarkemmin edellä mainitut käsitteet ja tarkastellaan niiden yhteyttä toisiinsa. Tarkasteluun on käytetty hyödyksi teoriatiedon ja lukion opetussuunnitelman perusteiden tarjoamia näkökulmia. Lukion opetussuunnitelman perusteisiin 2019 ja 2016 viitataan myöhemmin lyhenteillä LOPS2019 ja LOPS2015.

2.1 Matematiikan oppiminen

Matemaattinen ajattelu on termi, joka on laajasti käytössä matematiikan opetusta käsittelevissä artikkeleissa. Ei ole kuitenkaan olemassa yksimielisyyttä siitä, mitä kyseisellä termillä tarkoitetaan. Osa tutkijoista on sitä mieltä, että matemaattinen ajattelu on ajattelua matematiikasta, kun taas toisten tutkijoiden mielestä se on matematiikan avulla tapahtuvaa ajattelua. On esitetty, että matemaattinen ajattelu voidaan yhdistää sekä loogiseen ajatteluun että luovaan ajatteluun, mutta samalla myös matemaattiseen ymmärtämiseen. [21, 58–59]

Matematiikan osaaminen voidaan jakaa viiteen komponenttiin, jotka yhdessä muodostavat yhtenäisen kokonaisuuden [9, s. 116]. Nämä komponentit ovat:

1. Käsitteellinen ymmärrys – ymmärrys matemaattisista käsitteistä, operaatioista ja laskutoimituksista.
2. Proseduraalinen sujuvuus – taito käyttää matematiikan eri menetelmiä joustavasti, oikeaoppisesti, tehokkaasti ja tarkoituksenmukaisesti.
3. Strateginen kompetenssi – kyky muodostaa, esittää ja ratkaista matemaattisia ongelmia.
4. Mukautuva päättely – pystyvyys loogiseen ajatteluun, reflektointiin, selittämiseen ja todistamiseen.
5. Yrittelijäisyys – tapa nähdä matematiikka järkevänä, tarpeellisena ja arvokkaana sekä usko ahkeruuden vaikutukseen omassa osaamisessa.

Ongelmanratkaisu on menetelmä, jonka tarkoituksena on kehittää ongelmanratkaisijan itsenäistä ajattelua [21, s. 37]. On esitetty, että ongelmanratkaisussa yhdistyisivät kaikki viisi edellä mainittua matematiikan osaamisen komponenttia [9, s. 117–118]. Huomataan, että matematiikan osaaminen ja ongelmanratkaisutaidot voidaan siis nähdä vahvasti yhteydessä toisiinsa. Erityisesti matemaattisella ongelmanratkaisutaidolla tarkoitetaan kykyä ratkaista matemaattisia taitoja vaativia ongelmia erilaisten ratkaisumallien ja -strategioiden avulla [13, s. 374].

Opetuksessa on otettu käyttöön myös ongelmanratkaisuun pohjautuva käsite avoin ongelmanratkaisu. Avoimen tehtävän ratkaisussa oppijalla on mahdollisuus tuoda mukaan lisäoletuksia, sillä tehtävän alku- tai lopputilanne ei ole tarkasti määritelty. Tällöin oikeita vastauksia on useita ja ne voivat olla hyvinkin erilaisia. [21, s. 37] Hyvässä ongelmatehtävässä oppija joutuu tarkastelemaan tuttua asiaa uudessa valossa. Ideaalitilanteessa oppija on onnistunut muodostamaan selkeän loogisen kokonaisuuden matemaattisesta tiedosta ja näin ollen pystyy palauttamaan sen tarvittaessa mieleen. Tälläinen tilanne tulee usein esiin ongelmatehtävän parissa, jossa ratkaisun löytämiseksi on hahmotettava tilanne uudella tavalla. [13]

Ongelmanratkaisu mainitaan LOPS2019:ssa [19] opetusmenetelmänä ja oppimiskokemuksena. Sen lisäksi ongelmanratkaisutaitojen kehittämistä pidetään yhtenä oppimistavoitteena useissa eri oppiaineissa, kuten myös matematiikassa [19]. Yksi tapa parantaa ongelmanratkaisutaitoja on kehittää ohjelmoinnillista ajattelua, sillä ohjelmoinnillinen ajattelu voidaan nähdä ongelman ratkaisemiseen johtava ajatteluprosessina [7, s. 7].

2.2 Ohjelmointiosaaminen

Ohjelmointiosaaminen on laaja käsite, joka kattaa monipuolisesti ajattelua, päättelyä, suunnittelua ja toimimista digitaalisessa maailmassa. Näitä taitoja on tuotu esiin opetussuunnitelmissa jo varhaiskasvatussuunnitelman perusteista lähtien, ja peruskoulun jälkeen oppilailla tulisikin olla jo hyvä käsitys lyhyistä ohjelmakoodeista ja teknologian mahdollisuuksista. [2]

Ohjelmointiosaaminen voidaan jakaa kolmeen osa-alueeseen [2]:

1. ohjelmoinnillinen ajattelu,
2. tutkiva työskentely ja tuottaminen,
3. ohjelmoitujen ympäristöjen tunteminen ja niissä toimiminen.

Ohjelmointiosaamisen ensimmäinen kohta, ohjelmoinnillinen ajattelu, voidaan Järvenpään [7, s. 7] mukaan nähdä prosessina, joka on mahdollista jakaa kolmeen vaiheeseen:

1. Ongelman purkaminen osiin. Samalla tunnistetaan tehtävän piirteet, jotka ovat ratkaisun kannalta oleellisia ja voidaan pohtia voisiko osia purkaa edelleen pienempiin

vaiheisiin.

2. Ratkaisun vaiheiden muodostaminen.
3. Ratkaisun yleistäminen siten, että sitä voidaan käyttää kaikissa samankaltaisissa tehtävissä.

Järvenpää [7, s. 7] esittää myös esimerkin ohjelmoinnillisen ajattelun käyttämisestä yksinkertaisessa matemaattisessa ongelmassa: kokonaislukujen laskemisessa allekkain. Ensiksi luku puretaan ykkösiin, kymmeniin ja niin edelleen, kunnes koko luku on käsitelty. Tämän jälkeen luvut asetetaan allekkain yhteenlaskua varten. Ratkaisun vaiheet muodostetaan laskemalla yhteen ensin ykköset, sitten kymmenet ja jatkamalla näin edelleen. Ratkaisun yleistäminen perustuu tapaan merkitä yhdeksää suuremman summan kymmenet seuraavaan suurempaan kertaluokkaan. Näin ratkaisua voidaan soveltaa kaikkien kokonaislukujen yhteenlaskuun.

Esimerkiksi laskutoimitus $687 + 178$ voidaan ratkaista purkamalla kumpikin luku ykkösiin, kymmeniin ja satoihin, ja asettamalla samaa paikkamerkintää vastaavat luvut allekkain. Yhteenlasku suoritetaan aloittamalla laskutoimitus ykkösistä $7 + 8 = 15$ ja merkkaamalla saadun luvun kymmenet seuraavan kertaluokan päälle. Tämän jälkeen lasketaan yhteen kymmenet $1+8+7 = 16$ ja siirretään taas summan kymmenet seuraavaan kertaluokkaan. Lopuksi lasketaan yhteen sadat $1 + 6 + 1 = 8$ ja nähdään, että $687 + 178 = 865$. Siis

$$\begin{array}{r} {\scriptstyle 1\ \ 1\ \ } \\ 6\,8\,7 \\ +\ 1\,7\,8 \\ \hline 8\,6\,5. \end{array}$$

Ohjelmointiosaamisen toinen kohta, tutkiva työskentely ja tuottaminen, on liitettävissä termiin tutkiva oppiminen. Tutkiva oppiminen on ongelmanratkaisua pohjautuen kokeilemiseen ja omien ajatusten reflektoimiseen. Silfverberg [23, s. 398] esittää tutkimustehtävän asettelun seuraavanlaiseksi jatkumoksi, jossa oppija

1. laatii ohjeen mukaisen piirroksen, taulukon, kuvaajan, ... tai käyttää valmista materiaalia,
2. kokeilee tai tutkii materiaalia,
3. tekee päätelmän,
4. testaa päätelmän toimivuutta,
5. perustelee havaintojaan ja väitteitään.

Ohjelmoinnillisen ajattelun prosessissa, erityisesti siirryttäessä ohjelmointiin ja ohjelmointiosaamisen kolmanteen kohtaan, ongelmaksi voi muodostua ongelman purkaminen tarpeeksi pieniin osiin. Ihmisistä tuntuu usein turhalta määritellä kaikki vaiheet yksityiskohtaisesti ja pieniä vaiheita pidetään helposti itsestään selvinä. Tämä on kuitenkin ohjelmoin-

nin kannalta ratkaiseva vaihe, sillä tietokoneelle annettavissa ohjeissa tulee olla jokainen pieninkin vaihe, jotta tietokone tietää suorittaa ne. [7, s. 8]

2.3 Ohjelmoinnillinen ajattelu osana matematiikan oppimista

Matematiikan lähestyminen kokeellisesti on tehty koko ajan helpommaksi ja houkuttelevammaksi erilaisten teknologisten apuvälineiden ansiosta ja niiden avulla [23, 396–397].

Matematiikan opiskelussa käsitteiden oppimista voi helpottaa, jos esillä oleva käsite on mahdollista muuttaa ensin yksittäisistä toimenpiteistä koostuvaksi algoritmiksi, jossa käydään läpi kaikki kyseistä käsitettä määrittelevät ominaisuudet [29, s. 302]. Algoritmin avulla käsite voidaan nähdä johdonmukaisena rakenteena. Tossavainen ja Leppäaho [29, s. 302] esittävät esimerkkinä prosenttiosuuden laskemiseen käytettävän algoritmin seuraavasti:

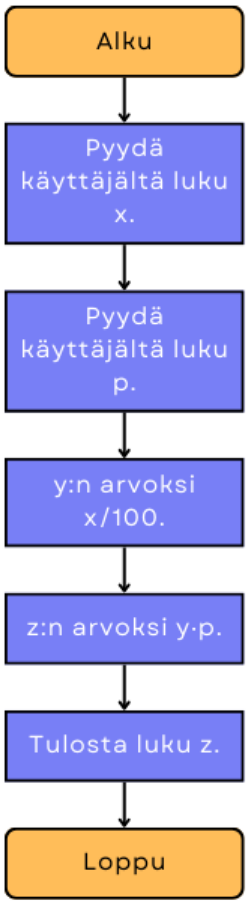
PROSEDUURI p_prosenttia_luvusta_x ALKAA

1. *Valitse luku x, josta prosenttiosuus lasketaan.*
2. *Valitse prosenttiluku p, jonka mukainen prosenttiosuus luvusta x lasketaan.*
3. *Jaa luku x sadalla ja merkitse tulosta kirjaimella y.*
4. *Kerro luku y luvulla p ja merkitse tulosta luvulla z.*
5. *Ilmoita vastauksena luku z.*

PROSEDUURI p_prosenttia_luvusta_x PÄÄTTYY

Kun opiskelija muodostaa muita prosenttilaskentaan liittyviä algoritmeja ja huomaa niiden välillä olevia yhteyksiä, jäsentyy yhteyksistä myöhemmin yhtenäinen kokonaisuus [29, s. 302]. Algoritminen ajattelu ei kuitenkaan ole itsestäänselvyys, vaan opeteltava taito [7, s. 12]. Yhdeksi apukeinoksi kehittää kyseistä taitoa on esitetty vuokaaviot, jotka ovat algoritmin graafisia esityksiä [7, s. 12]. Kuvassa 2.1 on Tossavaisen ja Leppäahon [29, s. 302] prosenttiosuutta kuvaavasta algoritmista muodostettu vuokaavio, joka esittää ohjelman rakenteen ideatasolla. Vuokaavioon on muokattu algoritmia siten, että ohjelman käyttäjän on mahdollista valita luku x ja luku p. Näin ohjelmaa on mahdollista käyttää yleisemmin kaikkia kokonaislukuja koskeviin prosenttiosuuksiin.

Algoritmit ja vuokaaviot toimivat oppimisen apuvälineinä, kun on aika siirtyä ongelman lausekieliseen ohjelmointiin. Ohjelma 2.1 on kirjoitettu Python-ohjelmointikielellä kuvan 2.1 vuokaavion avulla. Malliohjelma on voitu muodostaa suoraan vuokaavion askelten mukaisesti, mutta usein ohjelmakoodi vaatii vuokaaviota enemmän askelia toimiakseen. Näin on koottu yhtenäinen kokonaisuus prosenttiosuuden laskemisesta algoritmin ja vuokaavion avulla ja muokattu niistä toimiva ohjelmakoodi kyseiselle laskutoimitukselle.



Kuva 2.1. *Prosenttiosuuden laskemista kuvaava vuokaavio, joka on muodostettu Tossavaisen ja Leppäahon [29, s. 302] esittämästä algoritmista.*

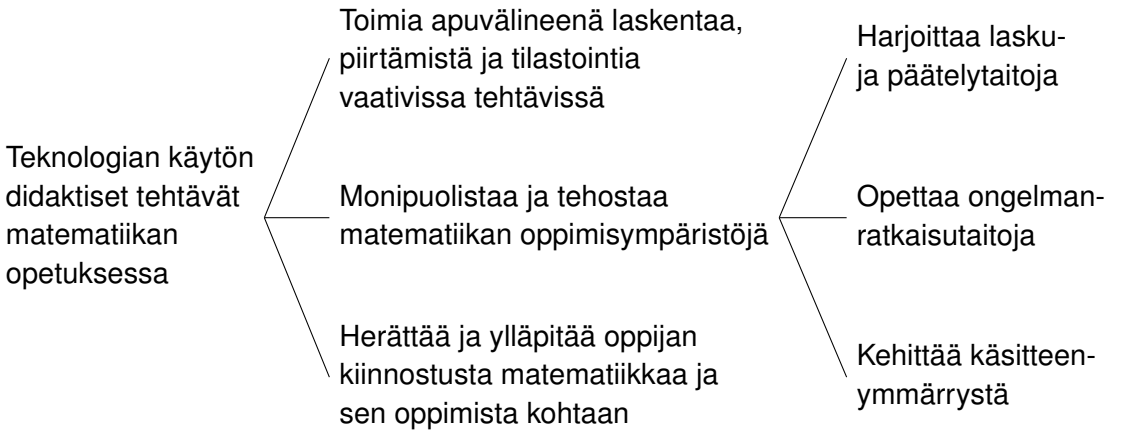
Ohjelma 2.1. *Kuvan 2.1 avulla muodostettu ohjelmakoodi Python-ohjelmointikielellä.*

```
1 x = int(input("Anna luku x, josta prosenttiosuus lasketaan: "))
2 p = int(input("Anna luku p, jonka mukainen prosenttiosuus "
3               "luvusta x lasketaan: "))
4 y = x/100
5 z = y*p
6 print("Prosenttiosuus on", z)
```

2.4 Teknologian hyödyntäminen matematiikassa

Viime aikoina on alettu korostaa tieto- ja viestintäteknologian merkitystä opetuksessa [12, s. 281][23, s. 395] ja tuotu esiin ajatusta siitä, miten lisääntynyt teknologia vapauttaa resursseja matemaattisen ajattelun ja ongelmanratkaisutaitojen edistämiseen [29, s. 303]. Lisäksi internetin avulla tapahtuvan oppimisen tarve on lisääntynyt koulu- ja oppilaitosverkoston supistumisen vuoksi [23, s. 396]. Tämä on johtanut siihen, että yhä useammin tietoa etsitään myös omatoimisesti muualta kuin perinteisistä oppimateriaaleista. Samaan aikaan tiedon oikeellisuuden merkityksen ja käyttökelpoisuuden arvioinnin vastuu on siirtynyt tiedon etsijälle ja käyttäjälle [23, s. 396]. Perusopetuksen opetussuunnitelman perusteissa 2014 [20, s. 22] painotetaankin monilukutaidon merkitystä, joka sisältää myös matemaattisen lukutaidon [23, s. 396].

Tieto- ja viestintäteknologian merkitystä matematiikan opetuksessa on kuitenkin tarkasteltu myös toisesta näkökulmasta. Silfverberg [23, s. 399] esittää, että teknologian lisääntyminen saattaa myös vaarantaa matemaattista ajattelua, jos teknologiaa käytetään hyödyksi ymmärtämättä matematiikan teoriaa sen takana. Hän osoittaa huolta esimerkiksi sovelluksia kohtaan, jotka hoitavat muun muassa yhtälöiden ratkaisemisen, derivoinnin ja integroinnin ja jättävät hyvin vähän tilaa käyttäjän matemaattiselle ajattelulle. Ihmisellä ja matematiikassa auttavalla koneella tuleekin olla hallittu vuorovaikutus, jossa ihminen osaa viestiä koneelle oikeassa muodossa, mutta samaan aikaan osaa myös arvioida saamaansa vastausta [23, s. 397]. Silfverberg [23, ss. 395 & 397] esittää teknologialle kuitenkin useita didaktisia tehtäviä matematiikan parissa (ks. kuva 2.2). Lisäksi hän tuo esiin virtuaalisten materiaalien hyvien puolia, joita ovat saatavuus ja kestävyys: materiaaleja riittää kaikille eivätkä ne kulu tai rikkoudu.



Kuva 2.2. *Teknologian tehtävät matematiikan opetuksessa. Kuva perustuu lähteeseen [23, s. 395].*

Tämän tukimateriaalin kannalta keskeisiä osia Silfverbergin jaottelussa ovat matematiikan oppimisympäristön monipuolistaminen sekä oppijan kiinnostuksen herättäminen ja ylläpitäminen matematiikkaa kohtaan. Digitalisaatiokehityksen eräs haaste on teknisten

apuvälineiden jatkuva ja nopea kehitys [23, s. 296]. Tämän tukimateriaalin tavoitteena onkin antaa yksi tapa lisää käyttää tietotekniikkaa apuna opetuksessa. Lisäksi tukimateriaalin tarkoituksena on opettaa ongelmanratkaisutaitoja ja kehittää käsitteenymmärrystä eli lisätä matemaattista lukutaitoa.

3. PYTHON-OHJELMOINTI

Tässä luvussa käsitellään sitä, miten ohjelmointi näkyy vuonna 2021 voimaan tulleessa lukion opetussuunnitelman perusteissa 2019 ja miksi tämän tutkielman opetusmateriaaliin on valittu juuri Python-ohjelmointikieli.

3.1 Ohjelmointi ja Python-ohjelmointikieli

Jokainen meistä on ohjelmoinut elämänsä aikana tietoisesti tai tiedostamattaan [7, s. 7]. Leikki-ikäinen noudattaa rakennuspalikoiden mukana tullutta rakennusohjetta ja vanhetessaan rakennusohjeet saattavat vaihtua huonekalujen kokoamisohjeisiin tai ruoanlaiton yhteydessä noudatettaviin resepteihin. Yhtälailla tietokone tarvitsee ohjeita toimiakseen, sillä se tekee vain tismalleen sen, mitä sen käsketään tekevän [7, s.7].

Eri ohjelmointikieliä on satoja ja niiden välillä valitseminen voi tuntua hankalalta. TIOBE on organisaatio, joka on erikoistunut ohjelmistojen laadun arviointiin ja seurantaan. Heidän tekemän listauksen mukaan Python on jo jonkin aikaa pysytellyt suosituimpien ohjelmointikielien kärjessä [28]. Python on yleisin ohjelmointikieli, jota opetetaan ohjelmoinnin peruskursseilla, ja se on laajasti käytössä myös ammattimaisessa sovellus- ja ohjelmistotuotannossa [7, s. 7]. Python ohjelmointikielestä julkaistiin ensimmäinen versio vuonna 1991 ja sen jälkeen sitä on päivitetty jatkuvasti [22]. Se on niin kutsuttu korkean tason ohjelmointikieli, joka on tarkoitettu käyttäjäystävälliseksi, sillä sitä on helppo lukea ja kirjoittaa [27]. Esimerkiksi lukion matematiikan ja luonnontieteen kehittämisverkosto Lukema [8] on suositellut valitsemaan Pythonin lukiossa käytettäväksi ohjelmointikieleksi.

3.2 Ohjelmoinnin oppiminen ja opettaminen

Järvenpää [7, s. 14] esittää ohjelmoinnin oppimisen eräänlaisena algoritmina. Hänen mukaansa se koostuu kynnyskohdista, joista jokainen tulee ratkaista ennen seuraavaan siirtymistä. Kynnyskohdat järjestyksessä ovat muuttuja, muuttujan asetus, muuttujan arvon kasvatus, ehtolause, ja- ja tai-operaatioiden ero, toistolauseen toiminta, aliohjelma, parametrien välitys aliohjelmalle, funktionaalinen aliohjelma, funktion kutsu sekä funktion arvon palautuminen kutsujalle. Järvenpään [7, s. 14] mukaan viimeisten kohtien ymmärtäminen osoittaa jo vakuuttavaa perusohjelmointitaitoa.

Matematiikan osa-alueita voidaan käsitellä monipuolisesti ohjelmoinnin avulla ja ohjelmointitehtävistä voi luontevasti luoda matematiikan perusosaamista vahvistavaa lisämateriaalia, mutta myös vaativampia lisäpulmia [7, s. 13]. Tärkeää on kuitenkin aloittaa tarpeeksi yksinkertaisista tehtävistä, jotta opiskelijalla säilyy tunne siitä, että hän osaa ja kykenee tekemään ohjelmointitehtäviä. Opiskelijan tulisi kuitenkin pitää mielessä, että virheensietokyky on tärkeä taito ohjelmoidessa ja sitäkin taitoa voi kehittää. [7, s. 11]

3.3 Ohjelmointi osana lukion opetussuunnitelman perusteita

Kansallisen koulutuksen arviointikeskuksen Karvin [16, s. 34] julkaiseman tutkimuksen tuloksista nähdään, että lukuvuonna 2020–2021 vain noin 3,3% tutkimukseen osallistuneiden lukioiden opiskelijoista osallistui jollekin ohjelmoinnin opintojaksolle. Arviossa oletetaan, että yksikään opiskelija ei olisi osallistunut useammalle kuin yhdelle ohjelmoinnin opintojaksolle. Todellisuudessa osallistuneiden määrä on vielä tuota pienempi, sillä todennäköisesti ainakin osa opiskelijoista osallistui useammalle ohjelmointikurssille kuin vain yhdelle [16, s. 34]. Ohjelmointiopetuksen puuttuminen huomattiin myös raportissa Digiajan peruskoulu II [25, s. 81], jossa haastatellut opettajat kertoivat ohjelmoinnin opetuksen haasteiksi muun muassa valmiiden materiaalien puuttumisen ja ajan puutteen matematiikan opetuksen ohessa. Lukion opetussuunnitelman perusteiden 2019 mukaan kaikkien lukioiden tulee tarjota ohjelmointia sisältävää pitkän matematiikan MAA11-opintojaksoa. Karvin tekemän tutkimuksen otantakoulujen rehtoreista noin joka kolmas ilmoittikin ohjelmoinnin opintojaksojen lisääntyvän tämän myötä. [16, s. 37]

Perusopetuksen opetussuunnitelman perusteissa 2014 [20, s. 235] ohjelmointi on tuotu osaksi matematiikan opetusta vuosiluokille 3–6. Lukion opetussuunnitelman perusteissa 2015 [18] ohjelmointia ei mainita vielä ollenkaan. LOPS2015:ssa [18, s. 129] mainitaan kuitenkin matematiikan yhteydessä useita esimerkkejä tietokoneohjelmistoista, kuten dynaamisen matematiikan ohjelmistot, tilasto-ohjelmistot ja digitaaliset tiedonlähteet. Lukion opetussuunnitelman perusteet 2019 [19] tulivat voimaan elokuussa 2021. Niiden mukaan matematiikkaa opiskellessa opiskelija kehittyy tietokoneohjelmistojen käytössä, oppii hyödyntämään niitä sekä arvioimaan tietoteknisten välineiden hyödyllisyyttä ja niiden käytön rajallisuutta.

LOPS2019:n [19] mukaan pitkään matematiikkaan kuuluu yhdeksän pakollista moduulia, joiden lisäksi opiskelijan on mahdollista suorittaa kolme valinnaista opintojaksoa. Valinnaisella opintojaksolla Algoritmit ja lukuteoria (MAA11) oppimistavoitteisiin kuuluvat algoritmit käsitteenä, niiden toiminta sekä yksinkertaisten algoritmien ohjelmointi [19, s. 229]. Luvussa 2 esitetään esimerkki yksinkertaisesta algoritmista, vuokaaviosta ja niistä muodostetusta ohjelmakoodista.

Valinnaisessa moduulissa Analyysi ja jatkuva jakauma (MAA12) yleiset oppimistavoitteet ovat syventää ymmärrystä analyysin peruskäsitteistä ja integraalilaskennasta, oppia muo-

dostamaan ja tutkimaan aidosti monotonisen funktion käänteisfunktiota sekä perehtyä jatkuvaan todennäköisyysjakaumaan ja normaalijakaumaan [19, s. 229]. LOPS2019 [19, s. 229] esittää yhdeksi MAA12-opintojakson oppimistavoitteeksi myös funktioiden sekä epäoleellisten integraalien tutkimisen käyttäen apuna erilaisia ohjelmistoja. Tässä tutkielmassa on tarkoituksena liittää Python-ohjelmointi osaksi myös MAA12-opintojaksoa tukimateriaalin avulla.

3.4 Ohjelmoinnin opetus MAA11-opintojakson oppimateriaaleissa

Lukion pitkän matematiikan oppikirjoissa Juuri – MAA11 algoritmit ja lukuteoria [5] ja Moodi – MAA11 Algoritmit ja lukuteoria [4] ohjelmoinnin opetus sijoittuu opintojakson loppupuolelle. Opintojakson alkuun on molemmissa oppimateriaaleissa sijoitettu logiikka ja lukuteoria. Kummassakin oppikirjassa käytetään ohjelmointikielenä Pythonia ja ne sisältävät samat ohjelmoinnin käsitteet hieman eri järjestyksessä. Molemmissa edellä mainituissa materiaaleissa ohjelmoinnin opetuksen pääpaino on muuttujien käsittelyssä, ehtolauseissa ja toistolauseissa. Kumpikin materiaali esittelee try except -rakenteen ja aliohjelmat lyhyesti aihetta käsittelevän tehtävän avulla. Moodi-kirjassa tehtävät koskien aliohjelmaa ja try except -rakennetta ovat merkitty haastaviksi tehtäviksi.

Moodi-oppimateriaali [4] etenee ohjelmoinnillisten asioiden parissa hieman nopeammin kuin Juuri-oppimateriaali [5]. Moodin materiaalissa käydään myös hieman perusteellisemmin läpi ohjelmointiin liittyvä käsite liukuluku, joka Juuri-oppimateriaalissa tulee esiin vasta viimeisen harjoitustyön ohella. Lisäksi eroavaisuutena on se, että Moodi-oppimateriaali käsittelee ehtolauserakenteen yhteydessä myös elif-komennon.

Internetistä löytyy MAA11-opintojaksolle suunnattua ohjelmoinnin oppimateriaalia melko paljon. Ilmaiseksi ja vapaasti saatavilla ovat esimerkiksi Tie koodariksi [26] ja Matematiikka omaan tahtiin [14] -sivustojen materiaalit. Opetushallitus on rahoittanut kummankin materiaalin tuottamista. Tie koodariksi -oppimateriaali perustuu lyhyisiin teoriaesimerkkeihin, joiden jälkeen opiskelija voi kokeilla esimerkkiä vastaavan ohjelmakoodin kirjoittamista suoraa verkkosivuilla. Opiskelijan ei tarvitse avata muita sovelluksia tai ladata mitään omalle koneelleen. Verkkosivut vaativat kuitenkin käyttäjätunnuksen luomisen ja sisään kirjautumisen. Materiaali koostuu Python-ohjelmointiin johdattelevasta osasta ja ohjelmointitehtäviä sisältävästä osasta, jossa opiskelijalla on mahdollisuus päästä kirjoittamaan ratkaisuja haastaviinkin matematiikan tehtäviin. Tehtävissä aiheita ovat muun muassa Monte Carlo -simulaatio ja RSA-salaus.

Matematiikka omaan tahtiin -verkkosivujen [14] oppimateriaalin alussa on matemaattista ja ohjelmoinnillista ajattelua kehittävistä osio, joka käsittelee algoritmisen ajattelun, vuokaaviot, konnektiivit ja totuusarvot, jaollisuuden, Eukleideen algoritmin sekä aritmetiikan peruslauseen. Näiden jälkeen materiaali sisältää Python-ohjelmointia käsittelevän osuuden. Ohjelmointiosion alussa esitetään Pythonin ja koodieditori Visual Studio Co-

den asentaminen. Tämän jälkeen ohjelmointiosuus jatkuu Python-ohjelmoinnin perusteilla, joita käsitellään teorian avulla ja matematiikka-aiheisilla tehtävillä.

4. KÄÄNTEISFUNKTIO JA SEN DERIVAATTA

Tässä luvussa tutustutaan käänteisfunktioon. Aluksi määritellään muutamia matematiikan käsitteitä, jotka oletetaan lukijalle jo tutuiksi. Tämän jälkeen perehdytään niiden avulla käänteisfunktioon ja sen derivaattaan. Esimerkkejä käänteisfunktiosta ja sen derivaatasta esitettään arkusfunktioiden avulla.

4.1 Esitiedot

Tässä alaluvussa määritellään käänteisfunktion tarkasteluun tarvittavia tietoja.

Esitietoihin on käytetty lähteinä teosta Analyysiä reaaliluvuilla [3, ss. 30-32 & 73–74] sekä Tampereen yliopiston opintojaksojen Analyysi A [10, s. 14] ja Analyysi B [11, s. 7] luentomonisteita.

Määritelmä 4.1. Funktio $f : A \to B$ on *injektio*, jos $f(a_1) = f(a_2)$, jos ja vain jos $a_1 = a_2$.

Määritelmä 4.2. Funktio $f : A \to B$ on *surjektio*, jos jokaista $b \in B$ kohti on olemassa vähintään yksi $a \in A$, jolle $f(a) = b$.

Määritelmä 4.3. Funktio $f : A \to B$ on *bijektio*, jos se on injektio ja surjektio.

Määritelmä 4.4. Olkoon $R \subset A \times B$ relaatio, jonka *käänteisrelaatio* on

$$R^{-1} = \{(b, a) : (a, b) \in R\}.$$

Nyt $R^{-1} \subset B \times A$.

Toisin sanoen, funktio on bijektiivinen, jos jokaista $b \in B$ kohti on olemassa täsmälleen yksi $a \in A$, jolle $f(a) = b$.

Määritelmä 4.5. Olkoon $\varepsilon > 0$. Pisteen $a \in \mathbb{R}$ *ε-ympäristö* $U_\varepsilon(a)$ on sellaisten lukujen $x \in \mathbb{R}$ joukko, joilla

$$|x - a| < \varepsilon.$$

Siis

$$U_\varepsilon(a) = \{x \in \mathbb{R} : |x - a| < \varepsilon\}.$$

Määritelmä 4.6. Funktion f *derivaatta* pisteessä x on

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h},$$

jos raja-arvo on äärellisenä olemassa. Tällöin sanotaan, että funktio f on derivoituva pisteessä x.

4.2 Käänteisfunktio

Tässä alaluvussa tutustutaan tarkemmin käänteisfunktion määritelmään. Lähteinä on käytetty teoksen Analyysiä reaaliluvuilla [3] sivuja 31–32 ja 83–84 sekä Tampereen yliopiston opintojaksojen Analyysi A [10, ss. 19 & 125–126] ja Analyysi B [11, s. 25] luentomonisteita.

Lause 4.7. Olkoon $f : A \to B$ funktio. Sen käänteisrelaatio f^{-1} on funktio, jos ja vain jos f on bijektio.

Siis, jos funktio $f : A \to B$ on bijektio, sillä on olemassa *käänteisfunktio* $f^{-1} : B \to A$, jolla pätee, että

$$f^{-1}(b) = a \iff f(a) = b.$$

Todistus. Olkoon $f : A \to B$ funktio ja oletetaan, että sen käänteisrelaatio $f^{-1} : B \to A$ on myös funktio.

Olkoon nyt $b \in B$, jolle $f^{-1}(b) = a$ ja $a \in A$. Koska f^{-1} on funktion f käänteisrelaatio, niin tällöin $f(a) = f(f^{-1}(b)) = b$. Siis f on surjektio.

Olkoot vielä $a_1, a_2 \in A$, joille pätee, että $f(a_1) = f(a_2)$. Koska f^{-1} on funktion f käänteisrelaatio, saadaan, että $f^{-1}(f(a_1)) = f^{-1}(f(a_2))$ ja edelleen $a_1 = a_2$. Siis f on injektio ja näin ollen bijektio.

Oletetaan nyt, että funktio f on bijektio. Koska f on surjektio, niin jokaiselle $b \in B$ on olemassa $a \in A$, joilla $f(a) = b$ ja koska f on injektio, niin a on yksikäsitteinen. Siis, koska $f(a) = b$, niin $f^{-1}(b) = a$ jokaisella $b \in B$, ja f^{-1} on funktio. □

Määritelmä 4.8. Olkoot $f : A \to \mathbb{R}$ funktio ja $a \in A$ sellainen piste, että $[a-r, a+r] \subset A$ jollakin $r > 0$. Funktio f on *jatkuva pisteessä* a, jos

$$\lim_{x \to a} f(x) = f(a).$$

Siis funktio f on jatkuva pisteessä a, jos jokaista $\varepsilon > 0$ kohti on olemassa sellainen $\delta > 0$, että

$$|f(x) - f(a)| < \varepsilon,$$

kun $|x - a| < \delta$.

Lause 4.9. Olkoon $\Delta \subset \mathbb{R}$ väli ja $f : \Delta \to f(\Delta)$ jatkuva funktio, joka on aidosti kasvava (vähenevä). Tällöin $f^{-1} : f(\Delta) \to \Delta$ on aidosti kasvava (vähenevä) ja jatkuva.

Todistus. Todistetaan tapaus, jossa f on aidosti vähenevä. Tapaus, jossa f on aidosti kasvava menee vastaavasti ks. [3, 83–84] ja [10, 125–126].

Osoitetaan ensin, että $f^{-1} : f(\Delta) \to \Delta$ on funktio, sen jälkeen todistetaan, että se on aidosti vähenevä ja vielä lopuksi osoitetaan, että f^{-1} on jatkuva funktio.

Lauseen 4.7 nojalla funktioksi osoittamiseen riittää todistaa, että f on bijektio. Koska funktio f on aidosti vähenevä, niin

$$x_1 \neq x_2 \implies f(x_1) \neq f(x_2)$$

kaikilla $x_1, x_2 \in \Delta$. Siis f on injektio. Toisaalta, $f : \Delta \to f(\Delta)$ on selvästi surjektio, koska sen arvojoukko on joukon Δ kuva. Siis f on bijektio ja $f^{-1} : f(\Delta) \to \Delta$ on funktio.

Todistetaan sitten, että f^{-1} on aidosti vähenevä. Olkoon $y_1, y_2 \in f(\Delta)$ ja $y_1 > y_2$. Koska f on bijektio, niin on olemassa sellaiset $x_1, x_2 \in \Delta$, että $f(x_1) = y_1$ ja $f(x_2) = y_2$ ja koska $y_1 > y_2$, niin $f(x_1) > f(x_2)$. Koska f on aidosti vähenevä, niin

$$f^{-1}(y_1) = x_1 < x_2 = f^{-1}(y_2).$$

Siis f^{-1} on aidosti vähenevä funktio.

Tarkastellaan vielä funktion f^{-1} jatkuvuutta. Olkoot $p = \inf\{f(x) : x \in \Delta\}$ ja $s = \sup\{f(x) : x \in \Delta\}$. Jos $p = s$, niin f^{-1} on vakiofunktio ja sen arvojoukko on $\Delta = \{x_0\}$. Siis funktion f määrittelyjoukko on $\Delta = \{x_0\}$ ja funktio f^{-1} on jatkuva.

Oletetaan, että f ei ole vakiofunktio ja osoitetaan, että funktio f^{-1} on jatkuva välillä $[p, s]$. Oletetaan vielä, että $y_0 \in (p, s)$. Nyt on siis olemassa $a, b \in \Delta$, joilla $f(a) = p$ ja $f(b) = s$. Koska f on bijektio, on olemassa $x_0 \in (a, b)$, jolla

$$f(x_0) = y_0 \qquad ja \qquad f^{-1}(y_0) = x_0. \tag{4.1}$$

Valitaan mielivaltainen $\varepsilon > 0$ ja oletetaan, että $\varepsilon < \min\{x_0 - a, b - x_0\}$, siis $U_\varepsilon(x_0) \subset (a, b)$. Koska f on aidosti vähenevä, niin määritelmän 4.8 nojalla

$$f(x_0 - \varepsilon) > y_0 > f(x_0 + \varepsilon).$$

Merkitään

$$\delta = \min\{y_0 - f(x_0 + \varepsilon), f(x_0 - \varepsilon) - y_0\}.$$

Tällöin $\delta > 0$ ja

$$(y_0 - \delta, y_0 + \delta) \quad \subseteq \quad (f(x_0 + \varepsilon), f(x_0 - \varepsilon)), \tag{4.2}$$

joten

$$\begin{aligned}
y \in U_\delta(y_0) \quad &\Leftrightarrow \qquad y \in (y_0 - \delta, y_0 + \delta) \\
&\overset{(4.2)}{\Rightarrow} \qquad y \in (f(x_0 + \varepsilon), f(x_0 - \varepsilon)) \\
&\overset{f^{-1}\text{aid. vähenevä}}{\Leftrightarrow} \quad f^{-1}(y) \in (f^{-1}(f(x_0 - \varepsilon)), f^{-1}(f(x_0 + \varepsilon))) \\
&\overset{f^{-1}\text{bijektio}}{\Leftrightarrow} \quad f^{-1}(y) \in (x_0 - \varepsilon, x_0 + \varepsilon) \\
&\overset{(4.1)}{\Leftrightarrow} \qquad f^{-1}(y) \in (f^{-1}(y_0) - \varepsilon, f^{-1}(y_0) + \varepsilon) \\
&\Leftrightarrow \qquad f^{-1}(y) \in U_\varepsilon(f^{-1}(y_0)).
\end{aligned}$$

Siis

$$f^{-1}(y) \in U_\varepsilon(f^{-1}(y_0)) \qquad \text{aina, kun} \qquad y \in U_\delta(y_0).$$

Nyt, koska jokaiselle $\varepsilon > 0$ on olemassa $\delta > 0$, jolla $f^{-1}(y) \in U_\varepsilon(f^{-1}(y_0))$ aina, kun $y \in U_\delta(y_0)$ on funktio f^{-1} jatkuva pisteessä y_0. □

Määritelmä 4.10. Olkoon $n \in \mathbb{N}$, $n \neq 0$ ja funktio $f(x) = x^n$. Funktion f käänteisfunktiota merkitään $f^{-1}(x) = \sqrt[n]{x}$ tai $f^{-1}(x) = x^{1/n}$. Funktion f määrittely- ja arvojoukko ovat $[x, \infty)$, kun n on parillinen ja kun n on pariton, määrittely- ja arvojoukko ovat molemmat $\mathbb{R}$.

4.2.1 Trigonometristen funktioiden käänteisfunktiot

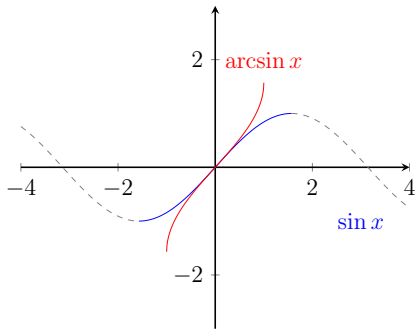
Tässä alaluvussa esitetään trigonometristen funktioiden käänteisfunktiot ja niiden kuvaajat. Myöhemmin luvussa 4.3.1 käsitellään kyseisten käänteisfunktioiden derivaattoja. Tämän alaluvun määritelmiin on käytetty lähteenä teosta Analyysia reaaliluvuilla [3, s. 146–148].

Määritelmä 4.11. Funktion $\sin : [-\frac{\pi}{2}, \frac{\pi}{2}] \to [-1, 1]$ käänteisfunktiota kutsutaan *arkussiniksi* ja sitä merkitään $\arcsin : [-1, 1] \to [-\frac{\pi}{2}, \frac{\pi}{2}]$.

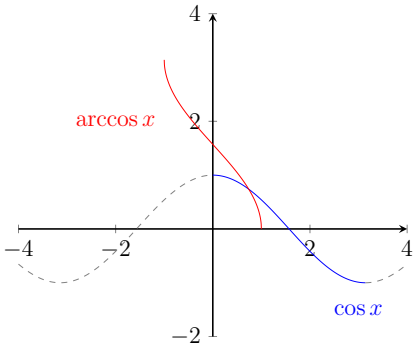
Määritelmä 4.12. Funktion $\cos : [0, \pi] \to [-1, 1]$ käänteisfunktiota kutsutaan *arkuskosiniksi* ja sitä merkitään $\arccos : [-1, 1] \to [0, \pi]$.

Määritelmä 4.13. Funktion $\tan : (-\frac{\pi}{2}, \frac{\pi}{2}) \to (-\infty, \infty)$ käänteisfunktiota kutsutaan *arkustangentiksi* ja sitä merkitään $\arctan : (-\infty, \infty) \to (-\frac{\pi}{2}, \frac{\pi}{2})$.

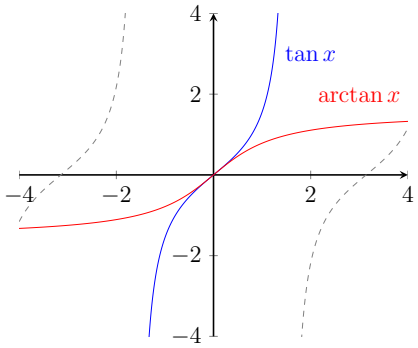
Arkusfunktioiden käänteisfunktioiden nimiin voidaan liittää myös tarkentava sana päähaara, joka viittaa funktioiden jaksollisuuteen.



(a) *Funktio* $\sin : [-\frac{\pi}{2}, \frac{\pi}{2}] \to [-1, 1]$ *ja sen käänteisfunktio* $\arcsin : [-1, 1] \to [-\frac{\pi}{2}, \frac{\pi}{2}]$.



(b) *Funktio* $\cos : [0, \pi] \to [-1, 1]$ *ja sen käänteisfunktio* $\arccos : [-1, 1] \to [0, \pi]$.



(c) *Funktio* $\tan : (-\frac{\pi}{2}, \frac{\pi}{2}) \to (-\infty, \infty)$ *ja sen käänteisfunktio* $\arctan : (-\infty, \infty) \to (-\frac{\pi}{2}, \frac{\pi}{2})$.

Kuva 4.1. *Trigonometristen funktioiden kuvaajat sinisellä ja niiden käänteisfunktioiden kuvaajat punaisella. Harmaat katkoviivat kuvaavat trigonometristen funktioiden jatkuvuutta, kun määrittelyjoukko laajennetaan kaikkien reaalilukujen joukkoon* $\mathbb{R}$.

Lause 4.14. Olkoot $x, y \in \mathbb{R}$, joille $xy < 1$. Tällöin

$$\arctan x + \arctan y = \arctan\left(\frac{x + y}{1 - xy}\right).$$

Todistus. Oletetaan, että lauseessa esitetyt oletukset pätevät.

Tarkastellaan ensin, mille välille luku $\arctan x + \arctan y$ kuuluu, kun $xy < 1$. Kun $xy < 1$ voi olla, että x ja y ovat erimerkkiset. Riittää kuitenkin tarkastella tapauksia $x > 0$, $x < 0$ ja $x = 0$.

Jos $x = 0$, niin

$$\arctan x + \arctan y = 0 + \arctan y = \arctan y.$$

Arkustangentin määritelmän nojalla $\arctan y \in (-\frac{\pi}{2}, \frac{\pi}{2})$. Siis $\arctan x + \arctan y \in (-\frac{\pi}{2}, \frac{\pi}{2})$, kun $x = 0$.

Jos $x > 0$ ja $xy < 1$ niin $y < \frac{1}{x}$. Merkitään nyt, että $x = \tan u$. Tällöin

$$\begin{aligned}\arctan x + \arctan \frac{1}{x} &= \arctan(\tan u) + \arctan(\frac{1}{\tan u}) \\ &= \arctan(\tan u) + \arctan(\tan(\frac{\pi}{2} - u)) \\ &= u + \frac{\pi}{2} - u \\ &= \frac{\pi}{2}.\end{aligned}$$

Arkustangentti on aidosti kasvava ja $y < \frac{1}{x}$, joten päätellään, että $\arctan x + \arctan y < \frac{\pi}{2}$. Lisäksi, kun $x > 0$, on $\arctan x > 0$ ja määritelmän perusteella $\arctan y > -\frac{\pi}{2}$, joten $\arctan x + \arctan y > -\frac{\pi}{2}$.

Siis, kun $x > 0$, niin $\arctan x + \arctan y \in (-\frac{\pi}{2}, \frac{\pi}{2})$.

Kun $x < 0$, saadaan vastaavanlaisella päättelyllä myös tulos $\arctan x + \arctan y \in (-\frac{\pi}{2}, \frac{\pi}{2})$.

Siis riippumatta muuttujan x merkistä, jos $xy < 1$, niin $\arctan x + \arctan y \in (-\frac{\pi}{2}, \frac{\pi}{2})$.

Olkoon $\arctan x = \alpha$ ja $\arctan y = \beta$. Tällöin $x = \tan\alpha$ ja $y = \tan\beta$, kun $\alpha, \beta \in (-\frac{\pi}{2}, \frac{\pi}{2})$. Nyt

$$\begin{aligned}\tan(\alpha+\beta) &= \frac{\sin(\alpha+\beta)}{\cos(\alpha+\beta)} \\ &= \frac{\sin\alpha\cos\beta + \cos\alpha\sin\beta}{\cos\alpha\cos\beta - \sin\alpha\sin\beta} \\ &= \frac{\frac{1}{\cos\alpha\cos\beta}(\sin\alpha\cos\beta + \cos\alpha\sin\beta)}{\frac{1}{\cos\alpha\cos\beta}(\cos\alpha\cos\beta - \sin\alpha\sin\beta)} \\ &= \frac{\frac{\sin\alpha\cos\beta}{\cos\alpha\cos\beta} + \frac{\cos\alpha\sin\beta}{\cos\alpha\cos\beta}}{\frac{\cos\alpha\cos\beta}{\cos\alpha\cos\beta} - \frac{\sin\alpha\sin\beta}{\cos\alpha\cos\beta}} \\ &= \frac{\frac{\sin\alpha}{\cos\alpha} + \frac{\sin\beta}{\cos\beta}}{1 - \frac{\sin\alpha\sin\beta}{\cos\alpha\cos\beta}} \\ &= \frac{\tan\alpha + \tan\beta}{1 - \tan\alpha\tan\beta} \\ &= \frac{x+y}{1-xy}.\end{aligned} \tag{4.3}$$

Kaavassa 4.3 voidaan laventaa lausekkeella $\frac{1}{\cos\alpha\cos\beta}$, sillä kun $\alpha, \beta \in (-\frac{\pi}{2}, \frac{\pi}{2})$, niin $\cos\alpha \neq 0$ ja $\cos\beta \neq 0$.

Ottamalla arkustangenttifunktion yhtälön molemmilta puolilta saadaan

$$\begin{aligned}\arctan\left(\tan\left(\alpha+\beta\right)\right) &= \alpha+\beta \\ &= \arctan x + \arctan y \\ &= \arctan\left(\frac{x+y}{1-xy}\right).\end{aligned}$$

On siis osoitettu, että väite pätee. □

4.3 Käänteisfunktion derivaatta

Tässä luvussa käsitellään käänteisfunktion derivoimiseen tarvittavia määritelmiä ja lauseita. Lähteenä tässä luvussa on käytetty kirjaa Analyysiä reaaliluvuilla [3, ss. 99, 107] ja Tampereen yliopiston opintojakson Analyysi B [11, s. 11] luentomonistetta.

Määritelmä 4.15. Olkoot A väli, $f : A \to \mathbb{R}$ funktio ja $x \in A$ piste. Oletetaan, että on olemassa $r > 0$, jolle $(x - r, x] \subset A$. Funktio f on *vasemmalta derivoituva* pisteessä x, jos on olemassa raja-arvo

$$\lim_{h\to 0^-} \frac{f(x+h)-f(x)}{h} \in \mathbb{R}.$$

Edellä esitettyä raja-arvoa kutsutaan funktion f *vasemmanpuoleiseksi derivaataksi* pisteessä x ja sitä merkitään $D_x^- f(x)$.

Määritelmä 4.16. Olkoon A, f ja x kuten määritelmässä 4.15. Oletetaan lisäksi, että on olemassa $r > 0$, jolle $[x, x + r) \subset A$. Funktio f on nyt *oikealta derivoituva* pisteessä x, jos on olemassa raja-arvo

$$\lim_{h\to 0^+} \frac{f(x+h)-f(x)}{h} \in \mathbb{R}.$$

Vastaavasti edellä esitettyä raja-arvoa kutsutaan funktion f *oikeanpuoleiseksi derivaataksi* pisteessä x_0 ja sitä merkitään $D_x^+ f(x_0)$.

Lause 4.17. Olkoon funktio $f : (a, b) \to \mathbb{R}$. Oletetaan, että f on derivoituva pisteessä $x \in (a, b)$, $f'(x) \neq 0$. Siis funktiolla f on käänteisfunktio f^{-1}, joka on määritelty välillä $(f(x) - r, f(x) + r)$ jollakin $r > 0$. Lisäksi funktio f^{-1} on jatkuva pisteessä $f(x)$. Tällöin funktio f^{-1} on derivoituva pisteessä $y = f(x)$ ja

$$(f^{-1})'(f(x)) = \frac{1}{f'(x)} = \frac{1}{f'(f^{-1}(y))}.$$

Todistus. [11, s. 25] Oletetaan, että lauseessa esitetyt oletukset ovat voimassa pisteessä x_0.

Koska f on derivoituva, se on jatkuva ja aidosti monotoninen jossakin pisteen x_0 ympäristössä. Tällöin funktiolla f on tällä välillä käänteisfunktio f^{-1}. Lisäksi nyt $x \to x_0$ täsmälleen silloin, kun $f(x) \to f(x_0)$ eli kun $y \to y_0$. Koska f on bijektio, voidaan vielä todeta, että $f(x) \neq f(x_0)$, kun $x \neq x_0$.

Siis

$$\begin{aligned}
\lim_{h\to 0} \frac{f^{-1}(y_0+h) - f^{-1}(y_0)}{h} &= \lim_{y\to y_0} \frac{f^{-1}(y) - f^{-1}(y_0)}{y - y_0} \\
&= \lim_{x\to x_0} \frac{f^{-1}(f(x)) - f^{-1}(f(x_0))}{f(x) - f(x_0)} \\
&= \lim_{x\to x_0} \frac{x - x_0}{f(x) - f(x_0)} \\
&= \lim_{x\to x_0} \frac{1}{\frac{f(x)-f(x_0)}{x-x_0}} \\
&= \frac{1}{f'(x_0)},
\end{aligned}$$

koska f on derivoituva pisteessä x_0 ja $f'(x_0) \neq 0$. Siis f^{-1} on derivoituva pisteessä y_0 ja

$$(f^{-1})'(y_0) = \frac{1}{f'(x_0)} = \frac{1}{f'(f^{-1}(y_0))}.$$

□

Esimerkki 4.18. [11, s. 27] Osoitetaan, että

$$D(\ln x) = \frac{1}{x} \qquad \text{kaikilla } x > 0.$$

Eksponenttifunktio e^x on aidosti kasvava ja jatkuva kaikilla $x \in \mathbb{R}$. Lisäksi $D(\mathrm{e}^x) = \mathrm{e}^x > 0$ kaikilla $x \in \mathbb{R}$. Siis eksponenttifunktiolla on käänteisfunktio $\ln x$, joka on lauseen 4.17 nojalla derivoituva kaikilla $x > 0$ ja

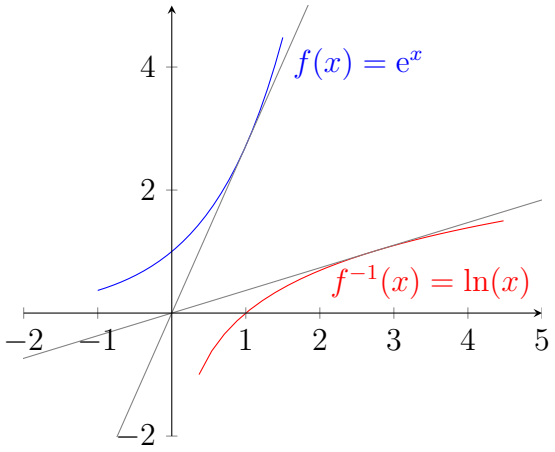
$$D(\ln x) = \frac{1}{D(\mathrm{e}^{\ln x})} = \frac{1}{\mathrm{e}^{\ln x}} = \frac{1}{x}.$$

Esimerkki 4.19. Olkoon funktio $f : (-1, \frac{3}{2}) \to \mathbb{R}, f(x) = \mathrm{e}^x$. Tarkastellaan funktion f derivoituvuutta pisteessä $x = 1$. Huomataan, että $f'(x) = \mathrm{e}^x$ ja $f'(1) = \mathrm{e} \neq 0$. Siis lauseen 4.17 nojalla funktiolla f on käänteisfunktio f^{-1}, joka on määritelty välillä

$(\mathrm{e}-r, \mathrm{e}+r)$ jollakin $r > 0$ ja joka on jatkuva pisteessä $f(1) = \mathrm{e}$. Siis funktio f^{-1} on derivoituva pisteessä $f(1)$ ja sen derivaatta on

$$(f^{-1})'(f(1)) = \frac{1}{f'(1)} = \frac{1}{\mathrm{e}}.$$

Kuvassa 4.2 on visualisoitu esimerkissä 4.19 kuvatut funktiot, sekä laskettujen derivaattojen avulla piirretyt tangentit.



Kuva 4.2. *Sinisellä piirretyn funktion* $f : (-1, \frac{3}{2}) \to \mathbb{R}, f(x) = \mathrm{e}^x$*, tangentti pisteessä* $x = 1$ *ja ja punaisella piirretyn käänteisfunktion* $f^{-1} = \ln(x)$ *tangentti pisteessä* $x = f(1) = \mathrm{e}$*.*

4.3.1 Arkusfunktioiden derivaatat

Esimerkki arkuskosinin derivaatasta mukailee opintojakson Analyysi B opintomonisteessa [11, s. 26] olevaa todistusta arkussinifunktion derivaatasta.

Esimerkki 4.20. Osoitetaan, että

$$D(\arccos x) = -\frac{1}{\sqrt{1-x^2}}.$$

Funktio $f(x) = \cos x$ on jatkuva ja derivoituva välillä $[0, \pi]$. Lisäksi $f'(x) = \sin x > 0$ kaikilla $x \in (0, \pi)$. Siis funktion f käänteisfunktio $f^{-1}(x) = \arccos x$ on lauseen 4.17 nojalla derivoituva kaikilla $x \in (0, \pi)$.

Tarkastellaan sitten arkuskosinin derivaattaa välillä $(0, \pi)$. Käytetään apuna trigonometrian peruskaavaa

$$\sin^2 \alpha + \cos^2 \alpha = 1,$$

josta laskettuna

$$\sin y = \pm\sqrt{1 - \cos^2 y}.$$

Merkitään $y = \arccos x$, jolloin $y \in [0, \pi]$. Koska sini on tällä välillä positiivinen, valitaan plusmerkki. Tällöin kaikilla $x \in (0, \pi)$ pätee

$$\sin(\arccos x) = \sqrt{1 - \cos^2(\arccos x)}.$$

Siis käänteisfunktion derivoimiskaavan perusteella, kun $y = \arccos x$, saadaan kaikilla $x \in (0, \pi)$

$$\begin{aligned} D(\arccos x) &= \frac{1}{D(f^{-1}(\arccos x))} \\ &= \frac{1}{D(\cos y)} \\ &= -\frac{1}{\sin y} \\ &= -\frac{1}{\sin(\arccos x)} \\ &= -\frac{1}{\sqrt{1 - x^2}}. \end{aligned}$$

Siis on osoitettu, että

$$D(\arccos x) = -\frac{1}{\sqrt{1 - x^2}}.$$

Esimerkki 4.21. Osoitetaan, että funktio

$$f(x) = \arctan x$$

on derivoituva. Määritelmän 4.6 nojalla funktio on derivoituva, jos raja-arvo

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

on äärellisenä olemassa.

Tarkastellaan ensin erotusosamäärän raja-arvoa, kun $x = 0$. Tällöin

$$\begin{aligned} f'(0) &= \lim_{h \to 0} \frac{f(0+h) - f(0)}{h} \\ &= \lim_{h \to 0} \frac{f(h) - 0}{h} \\ &= \lim_{h \to 0} \frac{f(h)}{h} \\ &= \lim_{h \to 0} \frac{\arctan h}{h} \\ &= 1. \end{aligned} \tag{4.4}$$

Todistetaan vielä kaavan 4.4 tulos $\lim_{x_0 \to 0} \frac{\arctan x_0}{x_0} = 1$.

Tiedetään, että $\lim_{x_0 \to 0} \frac{\sin x_0}{x_0} = 1$, joten

$$\begin{aligned}
\lim_{x_0 \to 0} \frac{\tan x_0}{x_0} &= \lim_{x_0 \to 0} \frac{\frac{\sin x_0}{\cos x_0}}{x_0} \\
&= \lim_{x_0 \to 0} \frac{\sin x_0}{x_0 \cos x_0} \\
&= \lim_{x_0 \to 0} \frac{\sin x_0}{x_0} \lim_{x_0 \to 0} \frac{1}{\cos x_0} \\
&= 1.
\end{aligned}$$

Merkitään $t = \arctan x_0$. Tällöin $x_0 = \tan t$ ja edelleen, kun $x_0 \to 0$ niin $t \to 0$. Siis soveltamalla l'Hospitalin sääntöä saadaan

$$\lim_{x_0 \to 0} \frac{\arctan x_0}{x_0} = \lim_{t \to 0} \frac{t}{\tan t} = 1.$$

Oletetaan seuraavaksi, että $x \in (0, \frac{\pi}{2})$ ja tarkastellaan erotusosamäärän raja-arvoa, kun $h \to 0$. Huomataan, että koska $h \to 0$, riittää seuraavassa päättelyssä tarkastella tilannetta, jossa $0 < x + h < \frac{\pi}{2}$. Tällöin $(x + h)(-x) < 0 < 1$, joten erotusosamäärän osoittajassa esiintyvälle arkustangenttien summalle voidaan käyttää lausetta 4.14.

Tarkastellaan siis funktion $f(x) = \arctan x$ raja-arvoa kun $h \to 0$. Huomataan, että

$$
\begin{aligned}
f'(x) &= \lim_{h\to 0} \frac{\arctan(x+h) - \arctan x}{h} \\
&= \lim_{h\to 0} \frac{\arctan(x+h) + \arctan(-x)}{h} \\
&= \lim_{h\to 0} \frac{1}{h} \arctan\left(\frac{x+h-x}{1-(x+h)(-x)}\right) \\
&= \lim_{h\to 0} \frac{1}{h} \arctan\left(\frac{h}{1+x(x+h)}\right) \\
&= \lim_{h\to 0} \frac{1}{h} \arctan\left(\frac{h}{1+x^2+hx}\right) \\
&= \lim_{h\to 0} \frac{1+x^2+hx}{h(1+x^2+hx)} \arctan\left(\frac{h}{1+x^2+hx}\right) \\
&= \lim_{h\to 0} \frac{1}{1+x^2+hx} \frac{1+x^2+hx}{h} \arctan\left(\frac{h}{1+x^2+hx}\right) \\
&= \lim_{h\to 0} \frac{1}{1+x^2+hx} \frac{\arctan\left(\frac{h}{1+x^2+hx}\right)}{\frac{h}{1+x^2+hx}} \\
&= \lim_{h\to 0} \frac{1}{1+x^2+hx} \cdot \lim_{h\to 0} \frac{\arctan\left(\frac{h}{1+x^2+hx}\right)}{\frac{h}{1+x^2+hx}} \\
&= \lim_{h\to 0} \frac{1}{1+x^2+hx} \cdot 1 \\
&= \frac{1}{1+x^2}.
\end{aligned}
$$

Vastaavasti voidaan käsitellä tapaus, kun $x \in (-\frac{\pi}{2}, 0)$. Tällöin $-\frac{\pi}{2} < x+h < 0$ ja edelleen $(x+h)(-x) < 0$, joten voidaan käyttää lausetta 4.14.

On siis osoitettu, että raja-arvo

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h} = \frac{1}{x^2+1}$$

on äärellisenä olemassa, ja funktio $f(x) = \arctan x$ on derivoituva.

5. TUKIMATERIAALIN SISÄLTÖ

Tässä luvussa käydään läpi tukimateriaalin tehtävät ja teoriaosuudet sekä esitetään perusteluja, miten ne tukevat oppimista. Luku etenee kronologisesti tukipaketin sisällön mukaan ja teksti on jaettu alalukuihin matemaattisten sisältöjen avulla.

Tukipaketin tehtävät on tarkoitettu tehtäväksi järjestyksessä, mutta opiskelijan on kuitenkin mahdollista jättää tehtäviä välistä esim. ajan puutteen vuoksi. Tehtävät voi ratkaista yksin tai ryhmässä. Niiden tekeminen vaatii matemaattista ymmärrystä, ei vain puhdasta ohjelmointitaitoa. Ohjelmointiin oletetaan käytettäväksi IDLE-kehitysympäristöä, sillä se asentuu valmiiksi Windows-tietokoneelle Pythonin asennuksen yhteydessä. Opiskelija voi kuitenkin valita minkä tahansa itselleen mieluisan ohjelmointiympäristön.

5.1 Yleisesti ohjelmointimateriaalista

Luvussa 3.2 esitettiin Järvenpään [7, s. 14] esittämät kynnyskysymykset ohjelmoinnin oppimiseen. Niistä muuttuja, asetuslause, muuttujan arvon kasvatus, ehtolause ja toistolauseen toiminta oletetaan opiskelijalle jo tutuksi MAA11-opintojaksolta sekä peruskoulun visuaalisesta ohjelmoinnista [1]. Nämä asiat kuitenkin kerrataan tukimateriaalissa muiden asioiden ohessa. Tukimateriaalin varsinainen teoriaosuus alkaa seuraavasta kynnyskysymyksestä eli ehtolauseiden käsittelystä. Teoria etenee melko vauhdikkaasti ja- ja tai-operaatioihin ja toistorakenteeseen, sillä nämä ohjelmoinnilliset käsitteet ovat olleet todennäköisesti esillä jo MAA11-opintojaksolla. Tukimateriaalissa olevien tehtävien kannalta nämä ovat kuitenkin erittäin tärkeitä asioita ja siksi niiden kertaus on perusteltua. Opiskelijalle todennäköisesti uusia asioita tukimateriaalissa ovat parametrien välitys aliohjelmalle, funktionaalinen aliohjelma, funktion kutsu sekä funktion arvon palautuminen kutsujalle.

Aliohjelmien toiminta on monille oppilaille vaikea asia ymmärtää. Aliohjelmat sisältävät osia ohjelman toiminnasta ja tekevät ohjelmasta helppolukuisen ja yleiskäyttöisen. Lisäksi aliohjelmat helpottavat ohjelman virheenetsintää ja ylläpitoa. Aliohjelmat voidaan kirjoittaa samaan tiedostoon pääohjelman kanssa tai erilliseen tiedostoon, jolloin ne tulee ottaa käyttöön Pythonin import-from -kutsulla. Jos aliohjelmat sijoitetaan samaan tiedostoon pääohjelman kanssa, tulee ne määritellä ennen kyseisen aliohjelman kutsumista. Aliohjelmia on mahdollista kutsua mistä tahansa Python-tiedostoista, jos ne on otettu tie-

dostoon käyttöön edellä esitetyn kutsun avulla ja siksi ne on järkevää kirjoittaa mahdollisimman yleiskäyttöisiksi. Yleiskäyttöisyyttä voidaan lisätä muun muassa aliohjelmien parametrien eli arvojen avulla. Parametrit annetaan aliohjelman käyttöön kutsun yhteydessä. Esimerkiksi eri tyyppisten parametrien avulla voidaan samalla aliohjelmalla lajitella järjestykseen sekä lukuja että nimiluetteloita. [7, s. 63]

Yleiskäyttöisyyttä voidaan lisätä myös Pythonin sisäänrakennetuilla kirjastomoduuleilla, jotka sisältävät toimintoja erilaisten tehtävien suorittamiseen. Kirjastojen toiminnot on usein toteutettu aliohjelmien avulla, joita kutsutaan tarvittaessa ohjelmakoodista. Myös kirjastot tulee ottaa käyttöön Pythonin import-kutsulla. Math-kirjasto on yksi tapa lisätä matemaattisia toimintoja Python-ohjelmointiin [7, s. 72] ja siksi se on käytössä myös tässä tukimateriaalissa. Math-kirjastoa ei tarvitse erikseen ladata omalle koneelle, joten sen käyttöönotto on yksinkertaista. Tukimateriaalista onkin haluttu rajata pois matemaattisia toimintoja sisältävät kirjastot, jotka vaativat erillisen lataamisen.

5.2 Epäoleellinen raja-arvo ja integraali

Tukipaketin alussa on lueteltu Python-ohjelmointikielen matematiikkamerkkejä ja math-kirjaston funktioita. Luettelot mahdollistavat tarvittaessa nopean tarkistuksen tietyille matematiikan merkille. Lisäksi ennen varsinaista teoriaosuuden alkua on kerrottu yleisiä vinkkejä helpottamaan Python-ohjelmointia, virheidenhallintaa ja IDLE-kehitysympäristön käyttöä. Näiden alkusivujen tarkoituksena on lisätä opiskelijalle hallinnan tunnetta ja rohkeutta aloittaa ohjelmointi. Hallinan tunne ja rohkeus on esitetty tärkeiksi tekijöiksi, jotka lisäävät opiskelijan mielikuvaa ohjelmoinnin hyödyllisyydestä [7, s. 11].

Tuttujen asioiden jälkeen teoriaosuus jatkuu aliohjelmilla. Aliohjelmat ovat todennäköisesti opiskelijalle vieraampi käsite, sillä ne eivät kuulu MAA11-opintojakson oppimistavoitteisiin. Aliohjelmat ovat suuressa roolissa tukimateriaalin tehtävissä, ja kun ne ovat esillä heti ensimmäisessä yksinkertaisessa tehtävässä, opiskelijalla on mahdollisuus kartuttaa itsevarmuuttaan ohjelmoinnin parissa. Tukimateriaalin ensimmäisen tehtävän ajatuksena on palauttaa mieleen Python-ohjelmoinnin syntakseja ja rakennetta. Oppimistavoitteena on erityisesti ymmärtää aliohjelmien sijainti ohjelmakoodin alussa, niihin siirtyminen pääohjelmasta ja palaaminen takaisin pääohjelmaan. Tukimateriaalissa kannustetaan sijoittamaan aliohjelmat heti tiedoston alkuun, jolloin ne ovat varmasti määriteltyjä silloin kun niitä tarvitaan.

Matematiikan osuus alkaa näkyä toisessa tehtävässä. Siinä palautetaan edelleen mieleen ehtolauserakennetta ja ohjelmoidaan kaksi yksinkertaista aliohjelmaa. Pääpaino ohjelmoinnin kannalta on ehtolauseissa ja kahden sisäkkäisen ehtolauseen muodostamisessa. Yksi haaste tehtävässä voi olla ehtolauseiden oikea sisentäminen. Matemaattinen sisältö on sijoitettu ehtolauseiden sisään ja korostaa matemaattista kielentämistä ja ymmärrystä. Tehtävä on mahdollista eriyttää ylöspäin antamalla vain matemaattiset lauseet

ja pyytämällä opiskelijaa luomaan oikeanlainen ehtolause niiden ympärille. Tällöin opiskelijan tulisi itse ymmärtää tarkastella nimittäjän ja osoittajan arvoja. Tehtävää voidaan käyttää MAA12-opintojaksolla raja-arvoa koskevien tapausten määrittelyyn tai se voi toimia kertaavana tehtävänä niiden käsittelyn jälkeen.

Kahden ensimmäisen tehtävän jälkeen teoriaosuus jatkuu math-kirjaston lyhyellä esittelyllä. Tehtävät 3–5 hyödyntävät math-kirjastoa ja käsittelevät MAA12-opintojakson aiheita raja-arvo äärettömyydessä, epäoleellinen integraali ja osissa laskettava epäoleellinen integraali. Tehtävissä 3 ja 4 hyödynnetään tutkivaa oppimista ja ongelmanratkaisua. Tehtävässä 3 opiskelija käyttää valmista ohjelmakoodia, tulkitsee sen antamia vastauksia ja muokkaa koodia vastaamaan tehtävänantoa. Tehtävässä 4 opiskelijalle on annettu valmis ohjelmakoodi, mutta sen rivit ovat väärässä järjestyksessä. Opiskelijan on tarkoitus tutkia, päätellä ja kokeilla, missä järjestyksessä rivien tulee olla, jotta ohjelmakoodi on toimiva.

Tehtävän 5 tarkoitus on mitata opiskelijan jo oppimaa ja samalla havainnollistaa sitä, miten erilaisten tehtävien ratkaisuun voidaan käyttää samoja ohjelmakoodeja. Tehtävä on siis mahdollista ratkaista tehtävän 1 ohjelmakoodilla, kun sen arvoja muokkaa tehtävän 5 mukaisesti. Ajatuksena on havainnollistaa ohjelmakoodien yleiskäyttöisyyttä ja kehittää ratkaisujen yleistämiseen liittyvää ohjelmoinnillista ajattelua.

5.3 Projekti funktion derivoinnista ja integroinnista

Kokoavan tehtävän 5 jälkeen tukimateriaalissa on teoriaosuus vuokaavioista, merkkijonoista ja listoista. Vuokaavion teoriaosuudessa on selitetty tässä materiaalissa käytettyjen vuokaaviomerkkien merkitykset ja kerrottu lyhyesti vuokaavioiden tulkitsemisesta. Opiskelijalle on annettu myös ajatusten herättelytehtävä, jossa tarkoituksena on tulkita vuokaavion avulla, mitä funktio tekee. Merkkijonoja esitellään rakennetasolla ja tämän jälkeen tuodaan esiin muutamia Pythonin valmiita merkkijonofunktioita. Opiskelijaa kehotetaan myös tarvittaessa etsimään lisää tietoa internetistä. Tämän teoriaosuuden lopuksi kerrotaan vielä lyhyesti listoista, niiden määrittelystä ja indeksoinnista.

Tehtävät 6, 7 ja 8 muodostavat yhden suuremman ohjelmakoodikokonaisuuden, jonka jälkeen tehtävän 8 ohjelmakoodia voidaan soveltaa tehtävissä 9, 10 ja 11. Tehtävissä 6–8 opiskelija tutustuu tarkemmin vuokaavioihin ja niiden tulkitsemiseen. Tehtävät on järjestetty niin, että haastavuus lisääntyy jokaisen tehtävän kohdalla. Tehtävä 6 harjoittaa ohjelmointitaitoja sekä merkkijonojen ja listojen käsittelyä. Tehtävä 7 syventää opittuja ohjelmointitaitoja, mutta vaatii jo enemmän sekä ohjelmoinnillista että matemaattista ymmärrystä. Opiskelijalle on esimerkiksi annettu vuokaaviossa ohjeeksi "laske derivaatta" ja hänen tulee osata muodostaa itse ohjelmakoodi muuttujista. Tehtävässä 8 muodostetaan uusi pääohjelma, jonka avulla liitetään tehtävät 6 ja 7 yhteen siten, että ohjelman avulla voidaan nyt laskea kokonaisen funktion derivaatta.

Tehtävän 8 jälkeen on bonustehtävä, jonka ajatuksena on tarjota haasteita ja pohdittavaa erityisesti heille, jotka ovat kokeneet edelliset tehtävät helpoiksi. Kohdan (a) toteuttaminen on kohtuullisen yksinkertaista ja sen tekeminen luo tehtävästä 8 käyttäjäystävällisemmän sekä antaa ohjelmalle lisää syvyyttä. Kohdassa (b) pyydetään opiskelijaa miettimään, miten ohjelmakoodia pitäisi muokata, jotta se toimisi halutulla tavalla. Matematiikan näkökulmasta voidaan huomata, että ohjelmaan on mahdollista syöttää funktio, joka sisältää negatiivisten termien yhteenlaskua ja tällä tavalla välttää suora vähennyslaskun luoma ongelma. Ohjelmoinnin näkökulmasta opiskelija huomaa varmasti useita ongelmakohtia, jos käytössä on vain tähän mennessä opitut taidot. Haasteina saattaa nousta esiin esimerkiksi se, miten funktio voidaan pilkkoa kahden eri merkin kohdalta sekä se, miten pilkkomisen jälkeen ohjelma saadaan muistamaan, oliko kyseessä positiivinen vai negatiivinen termi. Bonustehtävän kohta (b) pohjautuu avoimeen ongelmanratkaisuun ja tehtävän ratkaisuun on useita erilaisia näkökulmia ja vastauksia.

Tehtävä 9 mukailee tehtävän 5 ajatusta ja haastaa opiskelijan ymmärrystä siitä, mitä edellisissä tehtävissä on tehty. Tehtävän ratkaisu on kuitenkin yksinkertainen ja se voidaan muokata suoraan tehtävän 8 vastauksesta kahdella rivillä. Opiskelijalle annetaan mahdollisuus oivaltamiseen avoimen tehtävänannon avulla, mutta tarvittaessa opiskelijalle tarjotaan myös vinkki, joka kehottaa tarkastelemaan tehtävän 8 vastausta ja hyödyntämään sitä. Tehtävän ratkaisemisen jälkeen opiskelijalla on käytössä ohjelmakoodi, joka integroi saamansa funktion halutussa pisteessä. Tehtävässä 10 jatketaan integroinnin parissa. Tehtävänä on muokata koodia siten, että se integroi funktion määrätyllä välillä. Edelleen ohjelmoinnillisesta näkökulmasta tehtävä on melko yksinkertainen ja pääpaino on integraalilaskuun tarvittavien tietojen ymmärtämisessä. Opiskelijan tulee oivaltaa, mitä ohjelman täytyy tehdä, jotta kyseinen laskutoimitus on mahdollista suorittaa.

5.4 Jatkuvasti jakautuneen satunnaismuuttujan odotusarvo ja keskihajonta

Tehtävä 11 siirtyy käsittelemään jatkuvasti jakautuneen satunnaismuuttujan X odotusarvon laskemista tiheysfunktion avulla. Tämä tehtävä liittyy MAA12-opintojakson oppimistavoitteeseen koskien jatkuvaa jakaumaa. Tehtävänannossa on annettu algoritmi ohjelmakoodin kirjoittamista helpottamaan ja sen ensimmäisenä kohtana opiskelijaa pyydetään pohtimaan, mitä laskutoimituksia odotusarvon laskeminen vaatii ja miten sen voisi ratkaista kynän ja paperin avulla. Tämä johdattelee opiskelijan ajatuksia siihen, millaisia ohjelmakoodeja tehtävän ratkaisemiseen tarvitaan. Algoritmi on muodostettu siten, että se antaa ensin yleisen ohjeen ja tämän jälkeen opiskelija voi tarvittaessa lukea tarkemman ohjeen algoritmin alakohdista. Vaikka tehtävässä on annettu algoritmi avuksi, on se jo melko haastava ohjelmoinnin kannalta ja vaatii opiskelijalta enemmän ymmärrystä ja virheensietokykyä. Valmis ohjelmakoodi pää- ja aliohjelmoineen sisältää jo runsaasti rivejä.

Tästä syystä tukimateriaalissa käsitellään seuraavaksi aliohjelmien kutsumista toisesta tiedostosta.

Pitkien ohjelmatiedostojen välttämiseksi siirretään seuraavissa tehtävissä aliohjelmat toiseen tiedostoon, josta ne otetaan käyttöön Pythonin from-import -komennon avulla. Tämä saattaa myös lisätä opiskelijan ymmärrystä Python-kirjastojen toimintaperiaatteesta, sillä hän saa konkreettisen esimerkin kahden tiedoston välillä toimimisesta. Tehtävässä 12 opiskelija siis siirtää jo valmiin aliohjelman toiseen tiedostoon, tallentaa sen ja kutsuu sitä pääohjelmasta. Ajatuksena on varmistaa, että kutsuminen onnistuu ja perusajatus aliohjelman kutsumisesta toisesta tiedostosta on ymmärretty.

Tehtävä 13 jatkaa satunnaismuuttujien parissa toimimista, mutta ajatuksena on nyt kirjoittaa pääohjelma ja aliohjelmat omiin tiedostoihinsa, jotta yhden tiedoston pituus ei kasva liikaa. Tehtävän kohdissa (a) ja (b) opiskelija perehtyy tarkemmin jatkuvan satunnaismuuttujan keskihajonnan kaavan rakenteeseen ja siihen, mitä laskutoimituksia on mahdollista ratkaista jo valmiina olevien koodien avulla. Tämän jälkeen opiskelija ottaa käyttöön valmiiksi annetun ohjelmakoodin ja muokkaa sitä annettujen ohjeiden mukaisesti. Lopputuloksena ohjelman tulisi kyetä laskemaan keskihajonta annetun tiheysfunktion avulla.

Viimeinen tehtävä on materiaalin toinen bonustehtävä. Siinä opiskelijaa haastetaan pohtimaan global-muuttujan merkitystä valmiiksi annetussa ohjelmakoodissa. Opiskelijalla on mahdollisuutena myös etsiä tietoa internetistä ja kehittää monilukutaitoaan.

6. POHDINTA

Tässä luvussa käydään läpi tutkielman tekijän omia ajatuksia tuotetusta tukimateriaalista ja sen jatkokehityksestä. Lisäksi esitetään pohdintaa ohjelmoinnin opetuksesta ja Python-ohjelmoinnin käyttämisestä matematiikan opetuksessa.

6.1 Python-ohjelmoinnin mahdollisuudet matematiikan opetuksessa

Tukimateriaalin suunnitteluvaiheen alussa haasteeksi nousi yhtälöiden ratkaiseminen. Ei ollut yksinkertaista tapaa ratkaista yhtälöä ilman erikseen asennettavia Python-kirjastoja. Tässä tutkielmassa haluttiin kuitenkin välttää ylimääräisiä haasteita kirjastojen asentamisessa ja mahdollistaa tehtävien tekeminen mahdollisimman pienellä kynnyksellä. Päädyttiin siis ratkaisuun tehdä tehtäviä, joissa käsiteltiin funktioita merkkijonoina ja omien aliohjelmien avulla yhtälö voitiin ratkaista halutulla muuttujan x arvolla.

Python-ohjelmoinnin matemaattista sisältöä voi lisätä asentamalla tähän tarkoitettuja kirjastoja ja niiden avulla luoda lisää ulottuvuuksia ohjelmiin. Math-kirjaston lisäksi on mahdollista käyttää muitakin vapaan lähdekoodin kirjastoja, kuten NumPy ja Matplotlib. Matplotlib-kirjasto mahdollistaa tiedon visualisoinnin siten, että tehtyä tuotosta voidaan käyttää jopa web-julkaisuissa tai painotöissä [7, s. 79]. Sen avulla voi luoda visualisointeja kuten erilaisia graafeja sekä vuorovaikutteisia kuvia, joita on mahdollista zoomata ja päivittää [15]. NumPy-kirjastoa voidaan hyödyntää laskennallisissa tehtävissä [7, s. 79]. Se tarjoaa kattavia matemaattisia funktioita ja lineaarialgebrassa käytettäviä apuvälineitä [17]. Näiden lisäksi Python-ohjelmointiin on mahdollista ottaa käyttöön SymPy-kirjasto, joka mahdollistaa mm. yhtälönratkaisun, raja-arvolausekkeen sekä tarjoaa kombinatoriikan ja diskreetin matematiikan työvälineitä [24].

6.2 Ohjelmointiopetukseen varattava aika matematiikan opetuksessa

Tutkielmassa tuotettu tukimateriaali kattaa noin puolet MAA12-opintojakson keskeisestä sisällöstä. Opiskelijan on mahdollista tehdä materiaalin tehtäviä opintojakson aikana omaan tahtiin tai yhdessä matematiikan opetuksen kanssa. Koko opintojakson kattavan

ohjelmointimateriaalin luominen vaatisi oletuksen kattavammista pohjatiedoista, jotta teoriatietoa voitaisiin syventää entisestään ja tehtävien vaikeustasoa lisätä ilman, että opintojakson aikataulutus kärsisi. Kuten luvun 3 alaluvussa 3.3 esitettiin, opettajat näkevät ajan puutteen matematiikan opetuksessa usein syynä ohjelmointiopetuksen puuttumiselle.

Yhdeksi lukion MAA11-opintojakson keskeisistä sisällöistä on LOPS2019:ssa [19, s. 229] määritelty konnektiivit, joista ja- ja tai-konnektiivit vastaavat suoraa Pythonin and- ja or-operaatioita. On esitetty, että juuri nämä Boolen algebran and- ja or-operaatiot voivat olla oppilaille haastavia oppia [7, s. 87]. Olisi suhteellisen yksinkertaista luoda MAA11-opintojaksolle and- ja or-operaatioiden toimintaeroja havainnollistavia ohjelmointitehtäviä, joko konnektiivien läpikäynnin tai ehtolauseiden yhteydessä. Läpikäymissäni MAA11-opintojakson ohjelmointimateriaaleissa on kuitenkin hyvin vähän, jos lainkaan, aihetta käsitteleviä tehtäviä. Ja- ja tai-operaatioiden eron ymmärtäminen on joka tapauksessa oleellista ehtolauseita ja toistorakenteita käytettäessä, sillä ne mahdollistavat moniulotteisempia lauseita ilman sisäkkäisiä ehtoja. Jos tämä aihealue käytäisiin MAA11-opintojakson aikana läpi yhdessä konnektiivien ja totuusarvojen kanssa, yhdistyisivät ne suoraa matematiikan sisältöön ja uuden tiedon määrää vähenisi seuraavilla ohjelmointia sisältävillä kursseilla.

Jos opintojakson tuntisuunnitelma ei jaa tukimateriaalin sisältöä tasaisesti opintojakson ajalle, on suositeltavaa tehdä tukimateriaalin tehtäviä eri tahtiin tuntiopetuksen aiheiden kanssa. Esimerkiksi, jos käytössä on Juuri12 – MAA12 Analyysi ja jatkuva jakauma -oppikirjassa [6] oleva tuntisuunnitelma, ei tukimateriaalin tehtäville mahdollisesti jää tarpeeksi aikaa samaa aihetta käsittelevillä oppitunneilla. Edellä mainitun tuntisuunnitelman mukaisesti tukimateriaalin kymmenen ensimmäisen tehtävän tekemiseen olisi varattu seitsemäntoista 45 minuutin oppituntia tai kymmenen 75 minuutin oppituntia. Tämä tuntisuunnitelma sisältää tuntiopetuksen ja tunneilla käytössä olevan tehtävien tekoajan. Tämän jälkeen tukimateriaalin neljän viimeisen tehtävän tekemiseen jäisi kymmenen 45 minuutin oppituntia tai kuusi 75 minuutin oppituntia. Tukimateriaalin tehtäviä olisi tässä tapauksessa järkevää levittää koko opintojakson ajalle tasaisemmin.

6.3 Tukimateriaalin kehittämistä tukevat alku- ja loppukysely

Liitteenä olevien kyselyjen avulla materiaalia voi kehittää opiskelijoilta saadun palautteen perusteella. Ennen MAA12-opintojaksoa tehtävän kyselyn tarkoituksena on mitata opiskelijoiden lähtötasoa ohjelmointitaidoissa ja motivaatiota. Motivaatio voi vaikuttaa myös opiskelijoiden suoriutumiseen tukimateriaalin tehtävissä sekä tehtyjen tehtävien määrään. Nämä asiat huomioiden voidaan alkukyselyn tuloksia peilata loppukyselyn vastauksiin. Jos jonkin ohjelmointiaiheen haastavuus nousee erityisesti esiin alkukyselyssä, voi sitä kerrata lyhyesti yhdessä opiskelijoiden kanssa ennen tukimateriaalin käyttöönottoa.

Loppukyselyssä kannattaa erityisesti ottaa huomioon opiskelijoiden näkemys tehtävien

haastavuuteen ja teoriaosuuden selkeyteen. Jos tehtävät ovat tuntuneet erityisen helpoilta, voi kurssille kehittää lisämateriaalia kurssin loppupuolen asioista, joita tukimateriaalissa ei tällä hetkellä käsitellä, tai lisätä haastetta tehtäviin vähentämällä tai yksinkertaistamalla ohjeita. Vuokaaviotehtäviin saa helposti lisää haastetta yksinkertaistamalla toimintojen tekstejä ja jättämällä opiskelijalle enemmän tilaa omalle päättelylle. Jos taas tehtävät ovat tuntuneet haastavilta, voi materiaaliin lisätä esimerkkejä, visuaalista havainnollistusta tai useamman samantyylisen tehtävän, joista myöhempien tehtävien yhteydessä on mahdollista katsoa mallia.

LÄHTEET

[1] *ePerusteet. Digitaalisen osaamisen kuvaukset. Ohjelmoinnillinen ajttelu*. URL: https://eperusteet.opintopolku.fi/#/fi/digiosaaminen/8706410/osaamiskokonaisuuspaaalue/8715977 (viitattu 03. 12. 2023).

[2] *ePerusteet. Digitaalisen osaamisen kuvaukset. Ohjelmointiosaaminen*. URL: https://eperusteet.opintopolku.fi/#/fi/digiosaaminen/8706410/osaamiskokonaisuus/8709075 (viitattu 03. 12. 2023).

[3] P. Harjulehto, R. Klén ja M. Koskenoja. *Analyysiä reaaliluvuilla*. Gaudeamus Oy, 2014.

[4] J. Harsunkorpi ym. *Moodi. MAA11 Algoritmit ja lukuteoria*. Sanoma Pro Oy, 2021.

[5] M. Hähkiöniemi ym. *Juuri11 – MAA11 Algoritmit ja lukuteoria*. Otavan kirjapaino Oy, 2021.

[6] M. Hähkiöniemi ym. *Juuri12 – MAA12 Analyysi ja jatkuva jakauma*. Otavan kirjapaino Oy, 2022.

[7] J. Järvenpää. *Python-ohjelmoinnin opas opettajalle*. Otavan kirjapaino Oy, 2019.

[8] Lukema. Lukion matematiikan ja luonnontieteiden kehittämisverkosto. ”Ohjelmointi pitkässä matematiikassa – miten käytännössä?” (2019). URL: https://www.lukemaverkosto.fi/blogikirjoitukset/ohjelmointia-pitkassa-matematiikassa-miten-kaytannossa/ (viitattu 03. 11. 2023).

[9] J. Kilpatrick, J. Swafford ja B. Findell. *Adding it up: helping children learn mathematics*. National Academy Press, 2001.

[10] P. Koivisto. *Analyysi A*. Tampereen yliopisto, 2018.

[11] P. Koivisto. *Analyysi B*. Tampereen yliopisto, 2018.

[12] H. Krzywacki ja P. Portaankorva-Koivisto. ”Suomalainen matematiikan opettaja”. Teoksessa: *Matematiikan opetus ja oppiminen*. Bookwell Oy, 2018.

[13] H. Leppäaho. ”Ongelmanratkaisun opettamisesta”. Teoksessa: *Matematiikan opetus ja oppiminen*. Bookwell Oy, 2018.

[14] *Matematiikka omaan tahtiin. Algoritmit ja lukuteoria*. URL: https://matikka.omaantahtiin.com/pitkamatikka/maa11 (viitattu 16. 04. 2024).

[15] *Matplotlib: Visualization with Python*. URL: https://matplotlib.org/ (viitattu 03. 04. 2024).

[16] S. Nousiainen ja A. Kivistö. *Ohjelmoinnin opetuksen arviointi lukiokoulutuksessa*. Kansallinen koulutuksen arviointikeskus. 2022. URL: https://www.karvi.fi/fi/julkaisut/ohjelmoinnin-opetuksen-arviointi-lukiokoulutuksessa.

[17] *NumPy*. URL: https://numpy.org/ (viitattu 03. 04. 2024).

[18] Opetushallitus. *Lukion opetussuunnitelman perusteet 2015*. Opetushallitus, 2015. URL: https://www.oph.fi/fi/tilastot-ja-julkaisut/julkaisut/lukion-opetussuunnitelman-perusteet-2015.

[19] Opetushallitus. *Lukion opetussuunnitelman perusteet 2019*. Opetushallitus, 2019. URL: https://www.oph.fi/fi/tilastot-ja-julkaisut/julkaisut/lukion-opetussuunnitelman-perusteet-2019.

[20] Opetushallitus. *Perusopetuksen opetussuunnitelman perusteet 2014*. Opetushallitus, 2015. URL: https://www.oph.fi/sites/default/files/documents/perusopetuksen_opetussuunnitelman_perusteet_2014.pdf.

[21] E. Pehkonen ja M. Rossi. *Hyvää matematiikan opetusta etsimässä*. MFKA-Kustannus Oy, 2018.

[22] *python.org*. URL: https://www.python.org/ (viitattu 03. 11. 2023).

[23] H. Silfverberg. ”Tieto- ja viestintätekniikka matematiikan oppimisessa”. Teoksessa: *Matematiikan opetus ja oppiminen*. Bookwell Oy, 2018.

[24] *SymPy: Features*. URL: https://www.sympy.org/en/features.html (viitattu 03. 04. 2024).

[25] E. Tanhua-Piironen ym. *Digiajan peruskoulu II*. Opetus- ja kulttuuriministeriön julkaisuja 2020:17. 2020. URL: http://urn.fi/URN:ISBN:978-952-263-823-6.

[26] *Tie koodariksi. MAA11 Algoritmit ja lukuteoria*. URL: https://tie.koodariksi.fi/maa11/ (viitattu 16. 04. 2024).

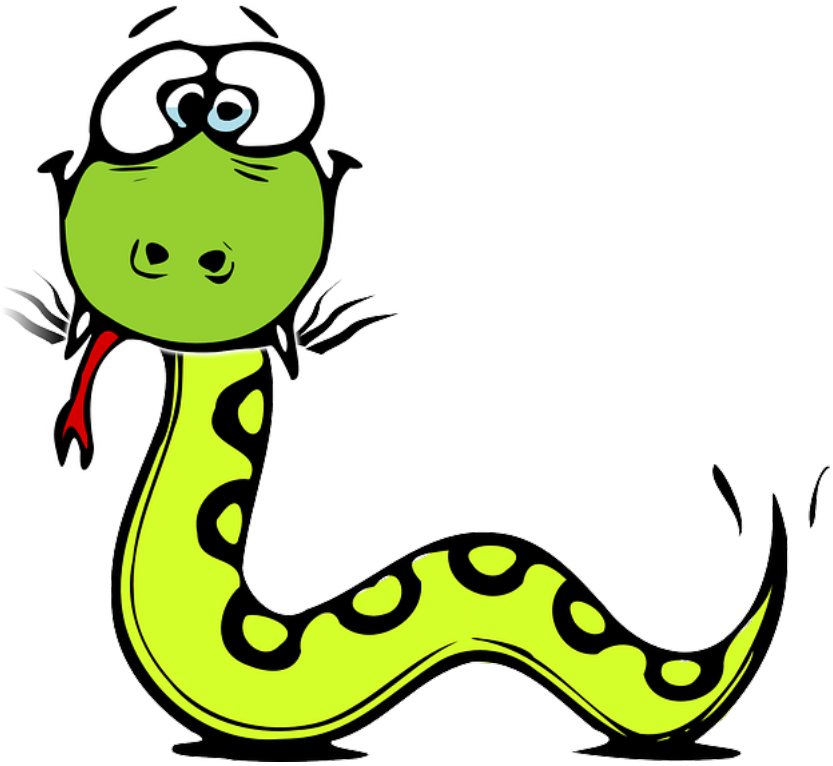
[27] Tieturi. ”Ohjelmointikielet. Ota ohjelmointikielet haltuun” (2023). URL: https://www.tieturi.fi/koulutusala/ohjelmistokehitys/ohjelmointikielet (viitattu 03. 11. 2023).

[28] *TIOBE the software quality company. TIOBE Index for October 2023*. URL: https://www.tiobe.com/tiobe-index/ (viitattu 03. 11. 2023).

[29] T. Tossavainen ja H. Leppäaho. ”Matematiikan opettajien ja opettajaksi opiskelevien matemaattisesta osaamisesta”. Teoksessa: *Matematiikan opetus ja oppiminen*. Bookwell Oy, 2018.

LIITE A: TUKIMATERIAALI

Matematiikan tehtäviä
Python-ohjelmoinnilla



Luettelo osasta Python-ohjelmointikielen matematiikkamerkeistä.

+	Yhteenlasku
-	Vähennyslasku
*	Kertolasku
**	Potenssiin korotus
/	Jakolasku
//	Osamäärän kokonaisosa
%	Jakojäännös
==	Yhtä suuri kuin
!=	Eri suuri kuin
<	Pienempi kuin
>	Suurempi kuin
<=	Pienempi tai yhtä suuri kuin
>=	Suurempi tai yhtä suuri kuin

Luettelo osasta Pythonin math-kirjaston funktioista.

math.sqrt(x)	Luvun x neliöjuuri
math.log(x)	Luvun x luonnollinen logaritmi
math.log(x, a)	Luvun x a-kantainen logaritmi
math.cos(x)	Kulman x kosini (x radiaaneina)
math.sin(x)	Kulman x sini (x radiaaneina)
math.inf	Ääretön
-math.inf	Miinus-ääretön
math.pi	π
math.e	Neperin luku e
math.exp(x)	e-kantaisen eksponenttifunktion arvo kohdassa x
math.ceil(x)	Kattofunktio luvusta x
math.floor(x)	Lattiafunktio luvusta x

A.1 Vinkkejä Python-ohjelmointiin

Muutama käytännön vinkki ohjelmakoodin kirjoittamiseen, virheiden välttämiseen ja niiden tulkitsemiseen.

A.1.1 Rivitys

Joskus ohjelmakoodia kirjoittaessa yhdelle komentoriville tuleva teksti venyy luettavuuden kannalta tarpeettoman pitkäksi. Komentoon liittyvän tekstin voi silloin jakaa kahdelle riville. Python jatkaa yhdellä rivillä alkavan komennon käsittelyä seuraavalle riville, kun komennossa on käytetty sulkuja niin, että aloittava sulje on ensimmäisellä rivillä ja lopettava sulje on seuraavalla rivillä. Kaarisulkeiden lisäksi sama toimii myös haka- ja aaltosulkeiden kanssa. Jos haluat jakaa print-komennon useammalle riville, lisää jokaisen tulostettavan rivin alkuun ja loppuun lainausmerkit.

```
print("Pitkä virke voidaan jakaa kahdelle riville, "
    "sillä virkettä ympäröivät kaarisulkeet.")
```

Jatkuvia rivejä kannattaa selkeyden vuoksi sisentää siten, että ne erottuvat muista riveistä.

```
if(luku >= 100
        and luku <= 1000):
    return luku
```

Pythonin omasta dokumentoinnista löytyy myös paljon muita tyylivinkkejä selkeään ja tyylillisesti oikeaan ohjelmointiin.[1] Tässä vaiheessa kannattaa kuitenkin keskittyä toimivan ja itselle riittävällä tasolla selkeän ohjelmakoodin kirjoittamiseen.

A.1.2 Tyyppimuutokset

Python-ohjelmoinnissa on mahdollista tehdä tyyppimuunnoksia lukuihin. Esimerkiksi kokonaislukutyyppi `int` on mahdollista muuttaa merkkijonoksi tai liukuluvuksi `float`, joka on tietokoneen tapa esittää reaaliluvut. Ohjelma ilmoittaa virheestä, jos se yrittää tehdä tyyppimuutosta väärän tyyppiseen muuttujaan. Siksi kannattaa olla tarkkana ja käyttää tyyppimuunnoksia vain silloin kun on varma käytössä olevan muuttujan tyypistä.

Käyttäjältä voidaan pyytää kokonaisluku:

```
luku = input("Anna kokonaisluku: ")
```

[1] PEP 8 – Style Guide for Python Code https://peps.python.org/pep-0008/

Ja saadaan merkkijonomuodossa (string) oleva luku. Esim. `"3"`. Jos `luku` halutaan muuttaa kokonaisluvuksi (integer), käytetään komentoa `int(luku)`. Tyyppimuunnos kokonaisluvuksi on mahdollista tehdä myös samalla, kun luku pyydetään käyttäjältä:

```
luku = int(input("Anna kokonaisluku: "))
```

Samaan tapaan muuttujaan `luku` syötetty kokonaisluku esim. `luku = 3` voidaan muuttaa takaisin merkkijonoksi komennolla `luku = str(luku)`. Ja edelleen liukuluvuksi käyttämällä komentoa `luku = float(luku)`. Näiden komentojen jälkeen komento `print(luku)` tulostaisi luvun `3.0`. Huomaa vielä, että liukuluvussa desimaalierottimena käytetään pistettä pilkun sijaan.

A.1.3 Rivinumerot

Saat rivinumerot näkyviin IDLE-kehitysympäristön tiedostossa valitsemalla Options → Show Line Numbers. Rivinumerot auttavat erityisesti virheilmoitusten tulkitsemisessa ja korjaamisessa.

A.1.4 Virheilmoituksen tulkitseminen

On hyvin tavanomaista, että koodia kirjoittaessa tekee virheitä. Virheiden korjaamista helpottaa, kun ne osaa paikallistaa. Lukemalla virheilmoitusta näet, missä tiedostossa ja millä rivillä virhe sijaitsee sekä millainen virhe on kyseessä. Kuvassa A.1 näkyy, miten IDLE-kehitysympäristö ilmoittaa virheestä tiedoston paaohjelma.py rivillä 56. Jos virheilmoituksessa näkyy useita rivejä, kannattaa tarkistaa ne kaikki.

```
Traceback (most recent call last):
  File "C:\Users\                        \Python\Mallikansio\paaohjelma.py"
, line 56, in <module>
    prin("Hello world!")
NameError: name 'prin' is not defined. Did you mean: 'print'?
>>>
```

Kuva A.1. *IDLE:n ilmoittama virhe tiedoston paaohjelma.py rivillä 56.*

Jos ohjelmakoodisi kääntyy, mutta ei tee sitä, mitä halusit, on erilaisia tapoja toimia: Jos ohjelma jää ikisilmukkaan eli ohjelma ei lopeta toimintaansa, vaikka odotat jonkin aikaa, voit itse lopettaa ohjelman painamalla näppäimiä ctrl + c samanaikaisesti. Jos ohjelma tulostaa väärän vastauksen tai muuten toimii omituisella tavalla, voit tehdä tarkistustulostuksia ohjelmakoodiin. Voit esimerkiksi tulostaa jokaisen aliohjelman paluuarvon ja tarkistaa, ovatko ne oikean tyyppisiä, ja onko niissä odotetut arvot. Tarkistustulostusten avulla näet helposti, missä kohdassa ohjelma ei toimi odotetulla tavalla.

A.2 Ehtolauseet

Python-ohjelmoinnissa ohjelman sisällä on mahdollista tehdä valintoja sen mukaan, täyttyvätkö halutut ehdot. Valinta tapahtuu `if, elif, else` -rakenteella.

Ohjelma A.1. *if, elif, else -toistorakenne*

```
1 if ehto:
2     toiminto
3 elif ehto:
4     toiminto
5 .
6 .
7 .
8 else:
9     toiminto
```

Ehtolauserakenteessa on aina tasan yksi `if`-lause. Pelkän `if`-lauseen avulla voit kirjoittaa ohjelmaan ehdon, joka toteutuessaan suorittaa halutun toiminnon ja jatkaa tämän jälkeen ohjelmaa. Jos ehto ei kuitenkaan toteudu, ohjelma ylittää `if`-lauseen ja jatkaa normaalisti etenemistä sen jälkeen.

Jos ohjelman rakenteessa on useita vaihtoehtoisia toimintoja, käytettään lisäksi `elif`-lausetta. Ehtolauserakenteessa voi olla yksi tai useampi `elif`-lause riippuen siitä, miten monta vaihtoehtoista tapahtumaa ohjelmassa on. Näistä ehdoista kuitenkin vain yksi toteutetaan. Ohjelma tarkistaa jokaisen `if/elif`-lauseen järjestyksessä ja etenee niistä ensimmäiseen, jonka ehto toteutuu. Tämän jälkeen ohjelma ylittää muut ehdot, vaikka ne toteutuisivatkin. Jos haluat, että ohjelma käy läpi kaikki toteutuvat ehdot, voit käyttää useita peräkkäisiä `if`-lauseita. Jos ohjelmassa ei ole yhtään toteutuvaa `if-` tai `elif`-lausetta, jatkaa ohjelma suoraa etenemistä niiden tarkistamisen jälkeen.

Jos ohjelmaan halutaan lisätä ehto, joka toteutetaan aina, kun `if-` tai `if, elif` -rakenne ei toteudu, lisätään ohjelmaan tasan yksi `else`-lause. Ohjelma toteuttaa `else`-lauseen sisältämän toiminnon, jos sitä edeltävät ehdot eivät toteudu. `Else`-lauseeseen ei kuitenkaan mennä, jos jokin sitä edeltävä `if/elif`-lause toteutuu.

A.2.1 Pythonin operaattorit

Ehtolauseita (ja toistolauseita) tehdessä kohdataan usein tilanne, jossa halutaan asettaa useampi ehto samaan lauseeseen. Tällöin voidaan käyttää `and-` ja/tai `or-` operaattoria. Nimensä mukaisesti `and`-operaattori toimii suomen kielen ja-sanan korvikkeena, kun taas `or`-operaattori toimii tai-sanan korvikkeena. Jos käytetään `and`-operaattoria, tulee sen yhteydessä olevien molempien ehtojen täyttyä, jotta ehto toteutuu. Kun käytetään

`or`-operaattoria, riittää, että toinen sen yhteydessä olevista ehdoista täyttyy, mutta myös molemmat ehdot voivat olla totta. Esimerkiksi ehtolause

```
if (luku > 5 and luku < 20):
```

on totta silloin kun luku kuuluu avoimelle välille $]5, 20[$. Sulkujen käyttö ehtolauseissa (tai toistolauseissa) ei aina ole pakollista, mutta usein se selkeyttää ja lisää luettavuutta. Lisäksi ne mahdollistavat ehtojen rivittämisen useammalle riville.

Pythonin `and`-/`or`-operaatioiden yhteydessä, jos ensimmäisen ehdon toteutuminen riittää määrittämään ehdon totuusarvon, ehdon loppuosaa ei koskaan suoriteta. Siis, jos ensimmäinen `or`-operaation ehdoista on totta, toisen ehdon totuusarvoa ei koskaan tarkisteta. Sama pätee, jos `and`-operaation ensimmäinen ehto on epätotta, toisen ehdon totuusarvoa ei tarkisteta.

Millainen luku toteuttaa seuraavan ehdon[2]:

```
if ((luku >= 2 and luku <= 10) or luku % 2 == 0):
```

[2] Luku, joka kuuluu suljetulle välille $[2, 10]$ tai on suurempi kuin 2.

A.3 Toistorakenteet

Python-ohjelmoinnissa on kaksi toistorakennetta: `for` ja `while`. Valinta näiden kahden väliltä perustuu usein siihen, tiedetäänkö miten monta toistoa tarvitaan. Toistorakenteesta voidaan käyttää myös nimityksiä luuppi (eng. loop) tai silmukka.

Komento `while` on usein käytössä silloin, kun ei tiedetä, miten monta kertaa ohjelma tulee suorittamaan toistorakenteen. Sen avulla voidaan määritellä, että ohjelma toistaa tiettyjä komentoja niin kauan kuin asetettu ehto on voimassa. Esimerkiksi ohjelma A.2 pyytää kokonaisluvun ja jos saatu luku on suurempi kuin nolla, ohjelma tulostaa sen, vähentää siitä yhden ja siirtyy uudelleen tarkastelemaan ehtoa. Näin jatkuu, kunnes vähennyksen jälkeen luku on yhtäsuuri kuin nolla.

Ohjelma A.2. *While-toistorakenne*

```
1 luku = int(input("Anna positiivinen kokonaisluku: "))
2 while luku > 0:
3     print(luku)
4     luku = luku - 1
```

Toistorakenne `for` on puolestaan yleensä käytössä silloin kun toistojen määrä on tiedossa etukäteen. Sen parametreiksi annetaan muuttuja sekä arvoväli, jonka muuttuja käy läpi. Ohjelma A.3 kuvastaa `for`-toistorakennetta, jossa muuttuja `i` käy läpi luvut välillä $0 - 10$ ja tulostaa niistä joka toisen. Muista, että range-funktion ylärajaa ei lasketa mukaan läpi käytäviin lukuihin.

Ohjelma A.3. *For-toistorakenne*

```
1 for i in range(0,10,2):
2     print(i)
```

Vastaavasti `for`-toistorakenteen avulla voidaan käydä läpi myös listoja tai tekstejä (jotka ovat kirjaimista koostuvia listoja). Ohjelma A.4 tulostaa erikseen jokaisen kirjaimen sanasta *matematiikka*.

Ohjelma A.4. *For-toistorakenne listalla*

```
1 for i in "matematiikka":
2     print(i)
```

A.4 Aliohjelmat

Python-ohjelmoinnissa on mahdollista luoda omia aliohjelmia, joita kutsutaan myös metodeiksi ja joskus funktioiksi. Ne tekevät ohjelmista helpommin luettavia ja muokattavia.

Aliohjelma voidaan kirjoittaa samaan tiedostoon pääohjelman kanssa ja ne sijoitetaan tiedostoon ennen niiden ohjelmakutsua. Selkein ratkaisu on sijoittaa aliohjelmat tiedostoon ennen pääohjelmaa. Silloin aliohjelmat on helppo löytää myöhemmin. Pääohjelmasta kutsutaan aliohjelmaa sen nimellä ja parametreilla. Kun pääohjelma etenee aliohjelman kutsuun, se siirtyy suorittamaan aliohjelman toiminnot ja palaa tämän jälkeen pääohjelmaan kutsua seuraavalle riville. Ohjelmassa A.6 on tiedoston alussa aliohjelma `yhteenlasku` ja tämän jälkeen pääohjelma. Ohjelman rakennetta on selitetty tarkemmin seuraavassa alaluvussa.

Aliohjelma kannattaa kirjoittaa siten, että sitä voidaan käyttää kaikkiin samantyyppisiin tehtäviin.

A.4.1 Aliohjelman rakenne

Aliohjelma alkaa sille varatulla sanalla `def`. Tämän jälkeen kirjoitetaan aliohjelman nimi ja sulkujen sisään sen parametrit eroteltuna pilkulla. Parametrilista voi olla myös tyhjä, `()`, jolloin aliohjelmalle ei anneta yhtään parametriä. Tämän jälkeen lisätään vielä kaksoispiste rivin päätteeksi. Seuraavat rivit sisennetään ja niissä tapahtuu itse aliohjelman toiminta.

Ohjelma A.5 toimii esimerkkinä aliohjelmasta, joka on kirjoitettu siten, että sillä on mahdollista laskea yhteen kahden luvun summa, mutta myös liittää yhteen kaksi merkkijonoa. Esimerkiksi komennolla `print(yhteenlasku(2, 3))` tulostuu luku `5` ja komennolla `print(yhteenlasku("ali", "ohjelma"))` tulostuu merkkijono `aliohjelma`.

Ohjelma A.5. *Yksinkertainen aliohjelma, joka laskee yhteen kaksi lukua ja palauttaa niiden summan.*

```
1 def yhteenlasku(x, y):
2     z = x + y
3     return z
```

Aliohjelman lopussa voi olla `return`-lause, joka palauttaa halutun arvon takaisin pääohjelmaan. Tämä ei ole kuitenkaan pakollista, vaan aliohjelma voi myös toteuttaa vain tietyn proseduurin ja palata sen jälkeen takaisin pääohjelmaan. Jos aliohjelma palauttaa arvon, tulee muistaa tallentaa saatu arvo pääohjelmassa tai käyttää sitä suoraan. Ohjelma A.6 tallettaa saadun summan muuttujaan `summa1` ja tämän jälkeen kutsuu aliohjelmaa uudelleen käyttäen tallennettua arvoa parametrina. Rivit 9–10 voidaan myös korvata yhdellä rivillä, jolla lukee `summa = yhteenlasku(yhteenlasku(x,y), yhteenlasku(x,y))`,

jolloin saatua paluuarvoa käytettään suoraa seuraavassa ohjelmakutsussa. Arvon tallentaminen toiseen muuttujaan voi kuitenkin joskus olla järkevää ohjelmakoodin luettavuun kannalta.

Mitä ohjelma A.6 lopuksi tulostaa?[3]

Ohjelma A.6. *Aliohjelma, jota kutsutaan kahdesti pääohjelmasta.*

```
1  def yhteenlasku(x, y):
2      z = x + y
3      return z
4
5  x = 2
6  y = 3
7
8  #lasketaan (x+y)+(x+y)
9  summa1 = yhteenlasku(x, y)
10 summa2 = yhteenlasku(summa1, summa1)
```

Edellä esitetyssä aliohjelmassa on käytetty muuttujia `x` ja `y`. Yksinkertaiseen ja helposti ymmärrettävään metodiin nämä sopivat, mutta muista nimetä muuttujat siten, että ohjelmakoodi on ymmärrettävää myös myöhemmin sekä sinulle että muille lukijoille. Muista myös, että voit aina lisätä selventäviä kommentteja koodiin #-merkkiä käyttämällä.

[3] 10

A.5 Tehtävät

1. Puuttuvat arvot

   Täydennä alla olevaan koodiin puuttuvat kohdat («???»), jotta ohjelma laskee ja tulostaa funktion

   $$f(x) = \begin{cases} -3x^2 + 3, & x < 1 \\ 3x, & x \geq 1 \end{cases}$$

   ja x-akselin välisen alueen pinta-alan välillä $[0, 3]$.

```
1  def f1(x):
2      #palauttaa funktion f(x) ensimmäisen osan integraalin
3      return <<???>>
4
5  def f2(x):
6      #palauttaa funktion f(x) jälkimmäisen osan integraalin
7      return <<???>>
8
9  #ensimmäisen osavälin alaraja
10 a = <<???>>
11
12 #ensimmäisen osavälin yläraja JA
13 #toisen osavälin alaraja
14 b = <<???>>
15
16 #toisen osavälin yläraja
17 c = <<???>>
18
19 A1 = f1(b)-f1(a)
20 A2 = f2(c)-f2(b)
21
22 print(A1+A2)
```

2. Osamäärän raja-arvo

   Tehtävässä käsitellään funktiota

   $$f(x) = \frac{3x}{(x-2)^2}.$$

   Kirjoita ohjelma, joka tekee kohdissa (a)-(d) pyydetyt asiat.

   (a) Tee oma funktio `osoittaja`, joka palauttaa funktion $f(x)$ osoittajan.

   (b) Tee oma funktio `nimittaja`, joka palauttaa funktion $f(x)$ nimittäjän.

   (c) Aseta muuttujalle x arvo $x = 2$.

   (d) Tarkasta, onko funktion $f(x)$ nimittäjä yllä asetetulla muuttujan x arvolla yhtä suuri kuin nolla. Käytä apuna edellä tekemääsi aliohjelmaa.

   Jos nimittäjän arvo on nolla, niin

   - tarkasta, onko osoittaja arvolla x suurempi kuin nolla.
     Jos on, tulosta "Funktiolla ei ole raja-arvoa kohdassa x =", x, ", mutta sillä on ainakin toispuoleiset epäoleelliset raja-arvot."
   - tarkasta, onko osoittaja arvolla x yhtä kuin nolla.
     Jos on, tulosta "Lauseketta on pyrittävä sieventämään, että raja- arvoista voidaan tehdä päätelmiä."

   Jos nimittäjän arvo ei ole nolla, niin tulosta "Funktiolle voidaan laskea raja-arvo."

   Aja ohjelma. Mitä ohjelma tekee?

A.6 Pythonin math-kirjasto

Matemaattisia funktioita Python-ohjelmointiin saadaan ottamalla käyttöön math--kirjasto. Kirjaston käyttöönotto tapahtuu lisäämällä tekstirivi `import math` ohjelman alkuun. Math-kirjasto mahdollistaa esimerkiksi funktioiden $\sin x, \cos x$ ja $\tan x$ käytön, mutta myös useita muita, joista osa on listattu materiaalin alussa. Pythonin math-kirjaston funktioita tulee kutsua lisäämällä alkuun `math.`, esimerkiksi `math.sqrt(x)` antaa luvun x neliöjuuren.

Jos tiedetään, että halutaan käyttää vain tiettyä kirjaston funktiota, on mahdollista ottaa käyttöön myös vain osa kirjastosta käyttämällä lausetta `from kirjastonNimi import haluttuOsa`. Esimerkiksi `from math import sin, cos`. Nyt math-kirjastosta ovat käytössä vain sini- ja kosinifunktiot. Tässä tapauksessa niitä voidaan käyttää suoraa muodossa `sin()` ja `cos()`.

A.7 Tehtävät

3. Funktion raja-arvo

   Ohessa oleva koodi laskee funktion $f(x) = x^4 + 2x^2 - 3$ raja-arvon, kun $x \to \infty$.

```
1  import math
2
3  #palauttaa polynomin f(x) korkeimman asteen termin,
4  #jota käytetään print-komennossa kertojana
5  def korkeinTermi(x):
6      return x**4
7
8  #palauttaa funktion, joka saadaan jakamalla f(x)
9  #korkeimman asteen termillään
10 def kerrottavaOsa(x):
11     return 1+2/x**2-3/x**4
12
13 print(korkeinTermi(math.inf)*kerrottavaOsa(math.inf))
```

   (a) Aja ohjelma. Mikä on funktion f raja-arvo, kun $x \to \infty$?

   (b) Tee ohjelma, joka laskee funktion $g(x) = x^3+3x^2+4$ raja-arvon, kun $x \to \infty$. Mikä on funktion g raja-arvo, kun $x \to \infty$?

   (c) Muuta ohjelmaa siten, että se laskee funktion g raja-arvon, kun $x \to 2$. Aja ohjelma. Mikä on nyt funktion g raja-arvo?

4. Kaavarivit solmussa

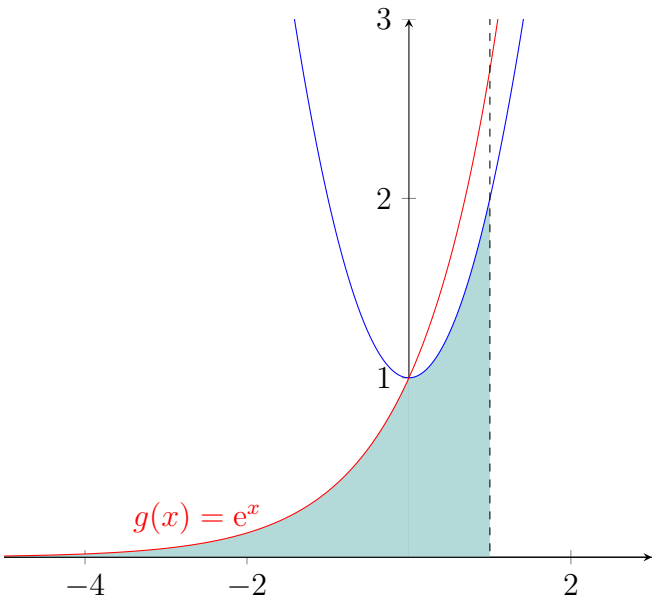
   Järjestä kaavarivit järjestykseen siten, että ne muodostavat järkevästi toimivan koodin.

```
def integroituf(x):
a = 4
return -6/math.sqrt(x)
integ = integroituf(t)-integroituf(a)
print(integ)
import math
t = math.inf
```

   Aja koodi. Minkä luvun koodi tulostaa? Miksi?

5. Funktion ja x-akselin välinen pinta-ala

   Kirjoita ohjelmakoodi, joka laskee funktioiden $g(x) = \mathrm{e}^x$ ja $f(x) = x^2 + 1$ sekä x-akselin rajaaman pinta-alan välillä $]-\infty, 1]$. Pinta-ala on esitetty väritettynä alla olevassa kuvassa.

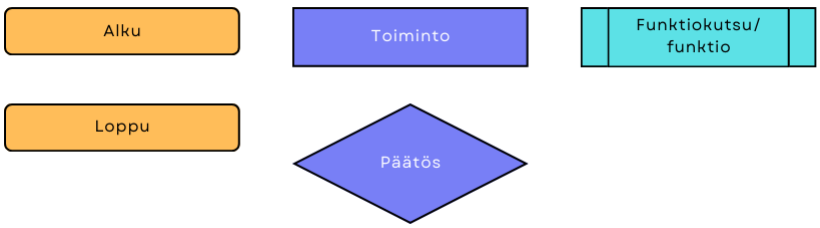


A.8 Vuokaaviot

Vuokaavio on graafinen tapa esittää algoritmeja. Se helpottaa algoritmin ymmärtämistä ja osoittaa selkeästi sen kulun. Vuokaavioilla voidaan esittää monimutkaisiakin algoritmeja siten, että lukijan on helppo seurata ohjelman etenemistä.

Vuokaaviot koostuvat symboleista, jotka kuvaavat algoritmin eri vaiheita. Kuvassa A.2 on esitetty viisi tässä materiaalissa käytössä olevaa symbolia.

- Algoritmin alku ja loppu on kuvattu keltaisella, pyöristetyllä suorakulmiolla.
- Toiminto, kuten muuttujan asettaminen tai muuttaminen, on kuvattu sinisellä suorakulmiolla.
- Päätös, kuten ehtolause tai toistorakenne, on kuvattu sinisellä neljäkkäällä.
- Toisen funktion kutsu ja sen alku on kuvattu vaaleansinisellä suorakulmiolla, jonka lyhyille sivuille on lisätty pystyviivat.



Kuva A.2. *Vuokaavion symbolit*

A.8.1 Lyhyt kertaus vuokaavion tulkitsemiseen

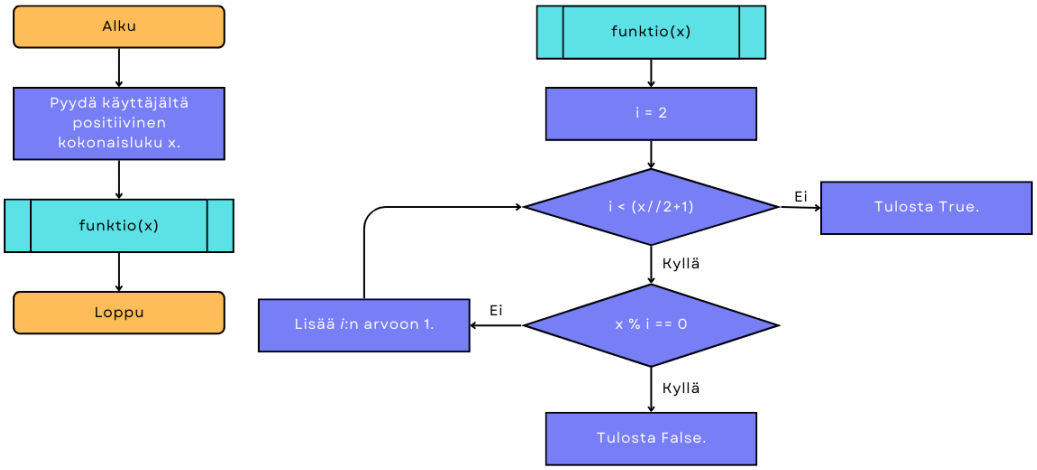
Vuokaaviot etenevät tavallisesti ylhäältä alas ja vasemmalta oikealle. Tilan puutteen vuoksi myös poikkeuksia voi olla.

Vuokaavion alkusymbolista lähtee yksi nuoli, joka osoittaa algoritmin seuraavan askeleen. Myös toiminto- ja funktiosymboleista lähtee tasan yksi nuoli. Päätössymbolista lähtee kaksi nuolta: vaihtoehdot kyllä ja ei. Päätössymbolin sisälle on tavanomaisesti kirjoitettu ehto, jonka toteutuessa siirrytään kyllä-nuolen suuntaan ja päinvastoin.

Funktiokutsusymbolista siirrytään ensin sitä vastaavaan funktiosymboliin ja toteutetaan kyseinen funktio loppuun. Tämän jälkeen jatketaan alkuperäisestä funktiosymbolista lähtevän nuolen osoittamaan suuntaan.

Ohjelma päättyy loppusymboliin, josta ei enää lähde nuolia eteenpäin. Loppusymboleita voi olla vuokaaviossa useampia, jos ohjelmalla on useita vaihtoehtoisia tapoja päättyä.

Kuvassa A.3 on vuokaavio, joka kuvaa erästä MAA11-opintojaksolla tutuksi tullutta matemaattista ilmiötä. Osaatko päätellä mitä? Voit katsoa vastauksen sivun alareunasta. [4]



Kuva A.3. *Vuokaavio mysteerifunktio*

Palauta mieleesi, miksi range-funktion ylärajaan lisätään $+1$. Muistatko mitä $\%$-merkki tekee Python-ohjelmointikoodissa?

[4] Vuokaavio kuvaa algoritmin, joka tarkistaa onko annettu luku alkuluku.

A.9 Merkkijonofunktiot

Merkkijono (eng. string) koostuu nimensä mukaisesti yksittäisistä merkeistä (eng. char) , jotka muodostavat jonon. Sana `"jono"` muodostuu siis neljästä merkistä `'j'`, `'o'`, `'n'` ja `'o'`. Pythonissa on mahdollista käyttää merkkijonoissa sekä puolilainausmerkkejä (') tai lainausmerkkejä ("). Tässä materiaalissa on käytetty lainausmerkkejä osoittamaan merkkijonoa ja puolilainausmerkkejä osoittamaan yksittäistä merkkiä. Tällainen merkintätapa on käytössä mm. Java- ja C++-ohjelmointikielissä. Voit valita itsellesi parhaan tavan käyttää lainausmerkkejä, mutta käytä niitä kuitenkin johdonmukaisesti.

Merkkijonoja voidaan käsitellä erilaisten merkkijonofunktioiden avulla. Merkkijonofunktiot eivät muuta merkkijonoa, vaan palauttavat arvon, joka voi olla merkkijono, lukuarvo tai totuusarvo. Ohessa on listattu muutama merkkijonofunktio, mutta niitä voi etsiä myös itse lisää internetistä käyttämällä hakusanoja *python string methods*.

len(m)	Palauttaa merkkijonon m pituuden. Siis, miten monta merkkiä jonossa on.
m.upper()	Palauttaa merkkijonon, jossa kaikki merkit ovat muutettu suuraakkosiksi (esim. jos m = "python", niin m.upper() palauttaa merkkijonon "PYTHON").
m.lower()	Palauttaa merkkijonon, jossa kaikki merkit ovat muutettu pienaakkosiksi (esim. jos m = "PYTHON", niin m.lower() palauttaa merkkijonon "python").
m.split(x)	Jakaa merkkijonon osiin niin, että jokainen jonossa m esiintyvä merkki x on katkaisukohtana. Palauttaa erotellut osat listaalkioina, jotka eivät sisällä kohdassa x annettua merkkiä. Esimerkiksi, jos m = "Python-ohjelmointi", niin m.split('n') palauttaa listan ["Pytho", "-ohjelmoi", "ti"].

A.10 Listat

Lista on tietotyyppi, joka yleensä sisältää useita alkiota, mutta voi myös olla tyhjä. Lista `luvut = [1, 2, 3, 4, 5]` on lista, jonka nimi on `luvut` ja se sisältää luvut 1–5. Lista `luvut2 = ["yksi", "kaksi", "kolme", "neljä", "viisi"]` on puolestaan lista, jonka nimi on `luvut2` ja se sisältää luvut yhdestä viiteen kirjoitettuna merkkijonoiksi. Listan arvoilla on jokaisella oma paikkaindeksi, joka Python-ohjelmoinnissa alkaa luvusta 0. Alkion arvo saadaan selville viittaamalla siihen: `luvut[2] = 3` tai `luvut2[0] = "yksi"`. Jos lista on tyhjä, sitä merkitään pelkillä hakasulkeilla ([ ]) eikä sen arvoihin silloin voi viitata paikkaindekseillä.

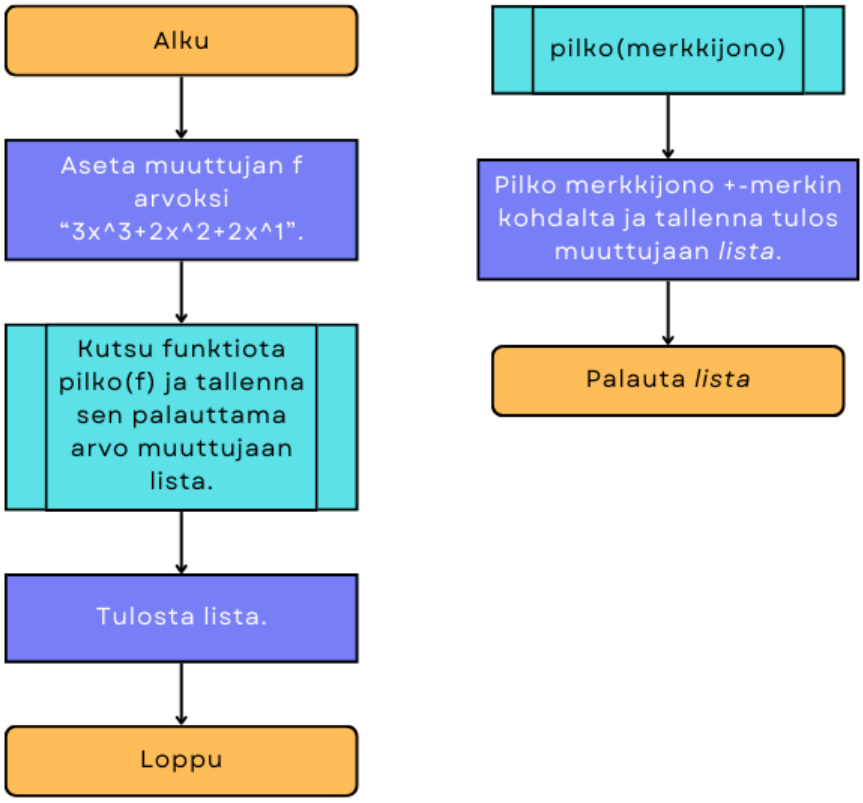
Listoilla, kuten merkkijonoillakin, on olemassa paljon omia listametodeja, jotka helpottavat niiden käyttöä ja muokkaamista. Tässä kohtaa kuitenkin riittää ymmärtää listan perusrakenne.

A.11 Tehtävät

Tehtävät 6–8 muodostavat yhden, suuremman ohjelman. Voit kuitenkin tehdä myös vain tehtävän 6 ja/tai 7, sillä niiden sisältämät ohjelmat toimivat myös omana ohjelmakoodinaan.

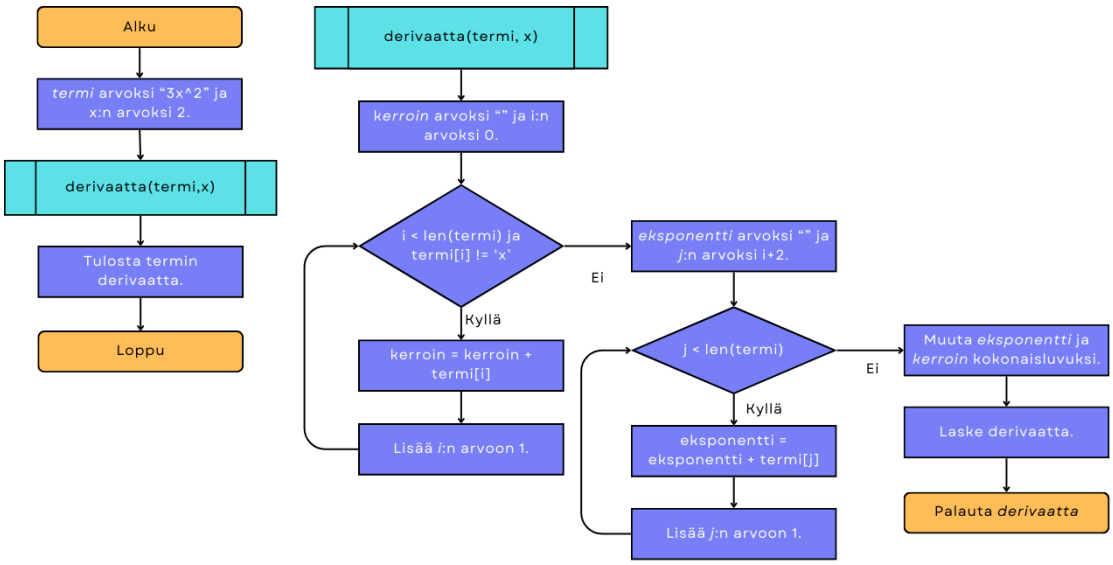
6. Pilko-aliohjelma

   Muodosta ohjelmakoodi, joka pilkkoo saamansa merkkijonomuotoisen funktion lausekkeen plusmerkkien kohdalta ja palauttaa listan funktion termeistä. Alla oleva vuokaavio kuvaa ohjelman toimintaa, voit käyttää sitä apuna koodin kirjoittamisessa.



7. Termin derivaatta

   Muodosta alla olevan vuokaavion kuvaama ohjelmakoodi, joka derivoi saamansa termin ja tulostaa lopuksi vastauksen. Toistorakenteissa käytetään muuttujia `i` ja `j`. Miksi muuttujan `i` arvoksi asetetaan `0` ja muuttujan `j` arvoksi asetetaan `i + 2`?



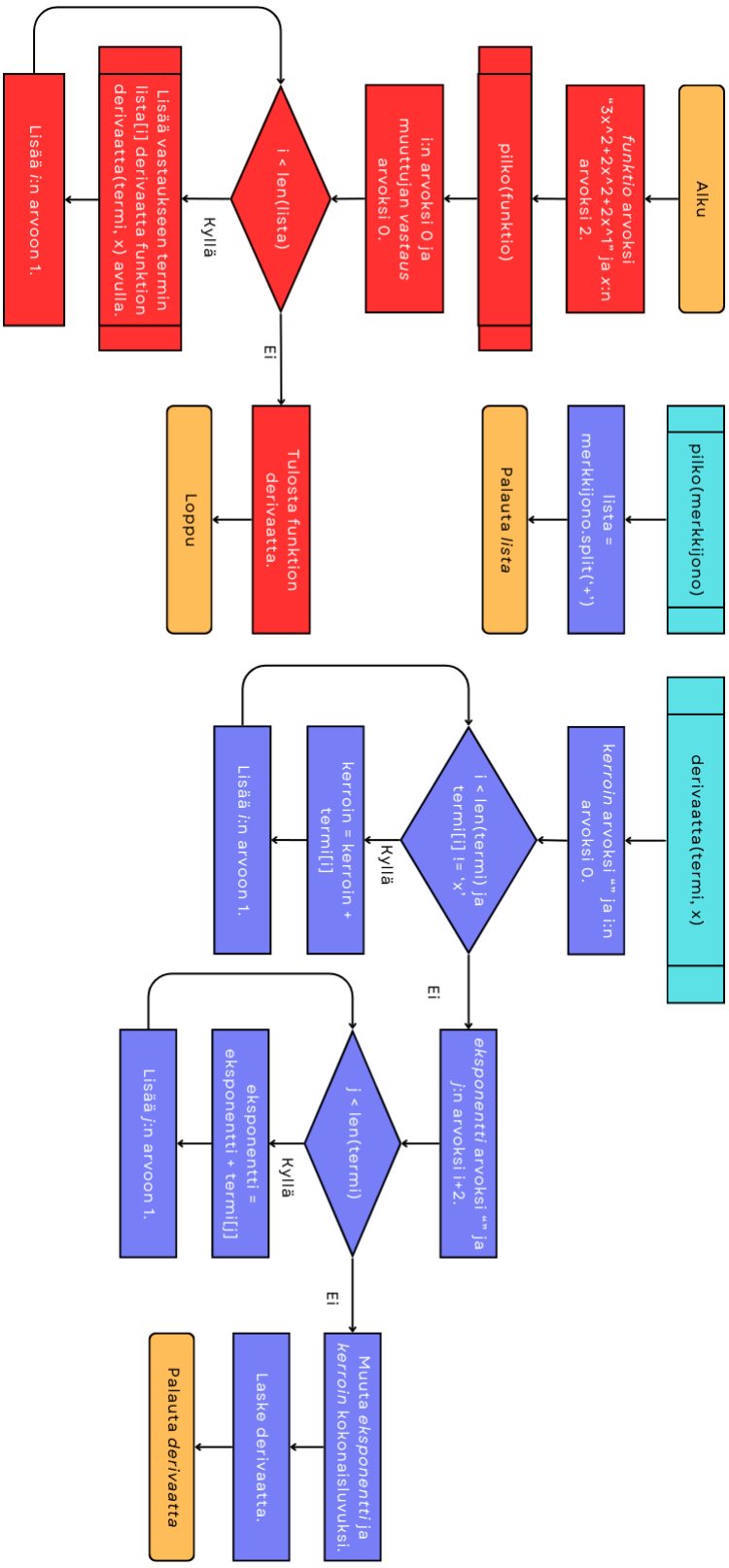
8. Funktion derivaatta

   Muodosta ohjelma, joka derivoi funktion $3x^2 + 2x^2 + 2x$. Aloita yhdistämällä tehtävien 6 ja 7 vastaukset. Voit käyttää apuna seuraavalla sivulla olevaa vuokaaviota. Siihen on merkitty punaisella muokatut kohdat. Huomaa, että mitään muokkauksia ei tarvitse tehdä tehtävissä 6 ja 7 tehtyihin funktioihin `pilko` ja `derivaatta`.

BONUSTEHTÄVÄ

(a) Muokkaa tekemääsi ohjelmakoodia siten, että käyttäjän on mahdollista antaa ohjelmalle oma funktio ja piste x.

(b) Mieti, mitä ohjelmakoodissa tulisi ottaa huomioon, jotta käyttäjän olisi mahdollista syöttää funktio, jossa on myös termien vähennyslaskua.


Alku
funktio arvoksi "3x^2+2x^2+2x^1" ja x:n arvoksi 2.
pilko(funktio)
i:n arvoksi 0 ja muuttujan vastaus arvoksi 0.
i < len(lista)
Kyllä
Lisää vastaukseen termin lista[i] derivaatta funktion derivaatta(termi, x) avulla.
Lisää i:n arvoon 1.
Ei
Tulosta funktion derivaatta.
Loppu
pilko(merkkijono)
lista = merkkijono.split('+')
Palauta lista
derivaatta(termi, x)
kerroin arvoksi "" ja i:n arvoksi 0.
i < len(termi) ja termi[i] != 'x'
Kyllä
kerroin = kerroin + termi[i]
Lisää i:n arvoon 1.
Ei
eksponentti arvoksi "" ja j:n arvoksi i+2.
j < len(termi)
Kyllä
eksponentti = eksponentti + termi[j]
Lisää j:n arvoon 1.
Ei
Muuta eksponentti ja kerroin kokonaisluvuksi.
Laske derivaatta.
Palauta derivaatta

9. Integraalifunktion laskeminen

   Tee ohjelmakoodi, joka integroi termien yhteenlaskua sisältävän funktion pisteessä x. Vinkki.[5]

10. Määrätty integraali

    Muokkaa tehtävän 9 ohjelmakoodia ja laske integraali $\int_1^5 (2x^3 - x)\,dx$. Jos et ole tehnyt tehtävää 9, voit kopioida koodin mallivastauksista. Vinkki.[6]

11. Odotusarvo

    Jatkuvan satunnaismuuttujan X tiheysfunktio on

$$f(x) = \begin{cases} -2x^2 + 4x, & \text{kun } 0 \leq x \leq 2 \\ 0, & \text{muualla.} \end{cases}$$

    Seuraavaksi tehdään ohjelmakoodi, joka laskee odotusarvon $E(X)$. Voit tehdä tehtävän ilman alla olevia ohjeita tai kohtien (a)–(e) avulla tai tarvittaessa katsoa lisäohjeita kohtien (a)–(e) alakohdista.

    (a) Mieti, millaisia laskutoimituksia ratkaisuun tarvitaan ja miten laskisit kyseisen satunnaismuuttujan odotusarvon kynän ja paperin avulla.

    (b) Hyödynnä edellisissä tehtävissä tekemiäsi aliohjelmia.

        - Tarvitset aliohjelman, joka pilkkoo merkkijonon ja aliohjelman, joka palauttaa saamansa termin integraalin halutussa pisteessä. Voit myös käyttää tehtävän 10 pääohjelmakoodia.

    (c) Jatkuvan satunnaismuuttujan odotusarvo voidaan laskea kaavalla

$$E(X) = \int_a^b x f(x)\,dx,$$

        kun tiheysfunktio f poikkeaa nollasta vain välillä $[a, b]$. Luo ohjelmakoodiin aliohjelma, joka saa parametrinaan alkuperäisen funktion termin ja palauttaa

[5] Mieti, voitko käyttää tehtävän 8 ohjelmakoodia hyödyksi. Tarvittaessa katso tehtävän 8 mallivastaus.

[6] Ohjelman täytyy laskea integraali kahdessa eri pisteessä ja tulostaa saatujen vastausten erotus.

termin niin, että muuttujan x eksponentti on kasvanut yhdellä. Esim. jos funktiolle annetaan parametrina termi $3x\hat{}2$ se palauttaa termin $3x\hat{}3$.

- Kopioi termin alkuosa `^`-merkki mukaan lukien toiseen muuttujaan (esim. `uusiTermi`).
- Kopioi termin eksponenttiosa toiseen muuttujaan (esim. `eksponentti`).
- Muuta eksponenttiosa numeeriseksi ja lisää siihen yksi.
- Lisää korotettu eksponenttiosa kopioidun termin loppuun merkkijonomuodossa ja palauta saatu arvo.

(d) Kirjoita pääohjelmaan toistorakenne, jonka avulla jokainen funktion termi käy kohdan (b) aliohjelmassa. Toistorakenne sijoittuu pääohjelmaan pilko-aliohjelman kutsun jälkeen.

- Toistorakenne on samanlainen kuin aiemmissa tehtävissä luodut while-toistorakenteet.
- Alusta ennen toistorakennetta uusi tyhjä lista.
- Lisää while-toistorakenteen sisällä aliohjelman palauttamat termit uuteen listaan.

(e) Huomaa, että pääohjelman tulee tämän jälkeen integroida muokattuja termejä, ei alkuperäisiä.

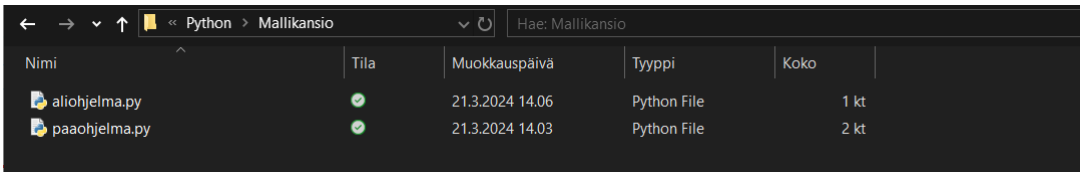
(f) Muokkaa ohjelman lopuksi tekemä tulostus siten, että se vastaa annettua tehtävänantoa.

Mikä on satunnaismuuttujan X odotusarvo $E(X)$?

A.12 Aliohjelman kutsuminen toisesta tiedostosta

Tehtävän 11 vastauksesta huomataan, että ohjelmakoodin luettavuus alkaa kärsiä, sillä tiedosto on jo melko pitkä. Ratkaisuna tähän on aliohjelmien tallentaminen erilliseen tiedostoon ja tiedoston ottaminen käyttöön `from`- ja `import`-komennoilla. Komennolla `import *` voidaan ottaa käyttöön kaikki tiedostossa olevat funktiot. Tämän jälkeen toisen tiedoston aliohjelmia voi käyttää samalla tavalla kuin aiemmin.

TÄRKEÄÄ! Tiedostojen tulee olla samassa hakemistossa (siis tallennettu samaan kansioon)! Katso kuva A.4.



Kuva A.4. *Kaksi Python-tiedostoa samassa kansiossa*

Nyt kuvan A.4 tiedostossa paaohjelma.py voidaan ottaa käyttöön tiedoston aliohjelma.py funktiot.

Ohjelma A.7. *Ohjelma, joka kutsuu aliohjelmaa toisesta tiedostosta.*

```
1 from aliohjelma import *
2
3 print(tervehdys())
```

Ohjelma A.8. *Aliohjelmatiedosto.*

```
1 def tervehdys():
2     return "Hei!"
```

A.13 Tehtävät

12. Kaksi tiedostoa

    Avaa tiedosto, johon olet kirjoittanut sekä pääohjelman että aliohjelman (esim. tehtävä 11). Siirrä aliohjelma omaan tiedostoon ja tee tarvittavat muutokset pääohjelmaan.

13. Keskihajonta

    Kun tiheysfunktio f poikkeaa nollasta vain välillä $[a, b]$, voidaan jatkuvan satunnaismuuttujan keskihajonta laskea kaavalla

    $$D(X) = \sqrt{\int_a^b (x - E(X))^2 f(x)\, dx}.$$

    (a) Mitä laskutoimituksia kaava sisältää?

    (b) Mitkä laskutoimitukset on mahdollista ratkaista jo tehtyjen ohjelmakoodien avulla?

    Kopioi seuraavien sivujen ohjelmakoodi omaan tiedostoon ja tallenna se omalle tietokoneellesi. Tiedosto kannattaa tallentaa valmiiksi samaan hakemistoon (kansioon) kuin missä edellisten tehtävien ohjelmakooditiedostot ovat.

    (c) Täydennä kopioimaasi koodiin puuttuva import-komento (<???>). Komennon avulla otetaan käyttöön aliohjelma, jonka avulla voidaan pilkkoa merkkijono plusmerkin kohdalta ja aliohjelma, jonka avulla voidaan integroida termi halutussa pisteessä. Huomaa, että tiedosto, josta aliohjelmat otetaan käyttöön, voi sisältää myös muita aliohjelmia.

    (d) Voit muokata ja käyttää tehtävän 11 pääohjelmakoodia.

        i. Ota pääohjelmassa käyttöön (c)-kohdassa tallentamasi tiedoston aliohjelmat.

        ii. Tallenna ohjelman laskema odotusarvo omaan muuttujaan.

        iii. Kutsu kopioimassasi koodissa olevaa keskihajonta-aliohjelmaa oikeilla parametreillä ja tallenna sen palauttama arvo eli satunnaismuuttujan X keskihajonta.

        iv. Tulosta vastaus.

(e) Olkoon erään jatkuvan satunnaismuuttujan X tiheysfunktio

$$f(x) = \begin{cases} -x^3 + 4x, & \text{kun } 0 \leq x \leq 2 \\ 0, & \text{muulloin.} \end{cases}$$

Määritä keskihajonta $D(X)$.

```
from <???> import *
from math import sqrt

def kerroin(termi):
    global i
    i = 0
    kerroin = ""
    while(termi[i] != 'x'):
        kerroin = kerroin + termi[i]
        i = i + 1
    return kerroin

def eksponentti(termi):
    global i
    i = i + 2
    eksponentti = ""
    while(i < len(termi)):
        eksponentti = eksponentti + termi[i]
        i = i + 1
    return eksponentti

def funktioidenTulo(f, g):
    fTermiLista = pilko(f)
    vastaus = ""
    j = 0
    while(j < len(fTermiLista)):
        ftermi = fTermiLista[j]
        fkerroin = kerroin(ftermi)
        feksponentti = eksponentti(ftermi)
```

```
        gTermiLista = pilko(g)
        k = 0

        while(k < len(gTermiLista)):

            gtermi = gTermiLista[k]
            gkerroin = kerroin(gtermi)
            geksponentti = eksponentti(gtermi)

            fgkerroin = (float(fkerroin) *
                         float(gkerroin))
            fgeksponentti = (float(feksponentti) +
                             float(geksponentti))
            vastaus = (vastaus + str(fgkerroin) +
                    "x^" + str(fgeksponentti) + "+")
            k = k + 1
        j = j + 1
    vastaus = vastaus[:-1]
    return vastaus

def keskihajonta(odotusarvo, tiheysfunktio, x, y):
    binominNelio = ("1x^2+-" + str(2*odotusarvo)
        + "x^1+" + str(odotusarvo**2) + "x^0")

    tulo = funktioidenTulo(binominNelio, tiheysfunktio)
    tuloTermit = pilko(tulo)
    i = 0
    vastaus1 = 0
    while(i < len(tuloTermit)):
        vastaus1 = (vastaus1 + integraali(tuloTermit[i], x))
        i = i + 1
    j = 0
    vastaus2 = 0
    while(j < len(tuloTermit)):
        vastaus2 = (vastaus2 + integraali(tuloTermit[j], y))
        j = j + 1

    keskihajonta = sqrt(vastaus2-vastaus1)
    return keskihajonta
```

BONUSTEHTÄVÄ Edellä olevassa ohjelmakoodissa on käytetty global-muuttujaa `i`. Miten se muuttaa ohjelmakoodin toimintaa?

A.14 Tehtävien vastaukset

1. Ohjelma laskee ja tulostaa funktion

$$f(x) = \begin{cases} -3x^2 + 3, & x < 1 \\ 3x, & x \geq 1 \end{cases}$$

ja x-akselin välisen alueen pinta-alan välillä $[0, 3]$.

```
1  def f1(x):
2      #palauttaa funktion f(x) ensimmäisen osan integraalin
3      return -x**3+3*x
4
5  def f2(x):
6      #palauttaa funktion f(x) jälkimmäisen osan integraalin
7      return (3*x**2)/2
8
9  #ensimmäisen osavälin alaraja
10 a = 0
11
12 #ensimmäisen osavälin yläraja JA
13 #toisen osavälin alaraja
14 b = 1
15
16 #toisen osavälin yläraja
17 c = 3
18
19 A1 = f1(b)-f1(a)
20 A2 = f2(c)-f2(b)
21
22 print(A1+A2)
```

2. Tehtävä voidaan ratkaista seuraavan koodin avulla.

```
def osoittaja(x):
    return 3*x

def nimittaja(x):
    return (x-2)**2

x = 2

if(nimittaja(x) == 0):
    if(osoittaja(x) > 0):
        print("Funktiolla f ei ole raja-arvoa kohdassa x = ",
              x, ", mutta sillä on ainakin toispuoleiset "
              "epäoleelliset raja-arvot.")
    elif(osoittaja(x) == 0):
        print("Lauseketta on pyrittävä sieventämään, että "
        "raja-arvoista voidaan tehdä päätelmiä.")
else:
    print("Funktiolle voidaan laskea raja-arvo.")
```

3. (a) Ohjelma tulostaa `inf`, joten raja-arvo on ∞.

   (b) Alla oleva koodi tulostaa luvun `inf`.

```
import math

#palauttaa polynomin f(x) korkeimman asteen termin,
#jota käytetään print-komennossa kertojana
def korkeinTermi(x):
    return x**3

#palauttaa muokatun funktion ilman erotettua kertojaa
def kerrottavaOsa(x):
    return 1 + 3/x + 4/x**3

print(korkeinTermi(math.inf)*kerrottavaOsa(math.inf))
```

(c) Alla oleva koodi tulostaa luvun 24.

```
1  import math
2
3  #palauttaa polynomin f(x) korkeimman asteen termin,
4  #jota käytetään print-komennossa kertojana
5  def korkeinTermi(x):
6      return x**3
7
8  #palauttaa muokatun funktion ilman erotettua kertojaa
9  def kerrottavaOsa(x):
10     return 1 + 3/x + 4/x**3
11
12 print(korkeinTermi(2)*kerrottavaOsa(2))
```

4. Tehtävään on useita erilaisia vastauksia. Alla yksi mahdollinen vastausvaihtoehto.

```
1  import math
2
3  def integroituf(x):
4      return -6/math.sqrt(x)
5
6  t = math.inf
7  a = 4
8  integ = integroituf(t)-integroituf(a)
9
10 print(integ)
```

Koodi tulostaa luvun 3. Se on funktion $\frac{3}{x\sqrt{x}}$ epäoleellinen integraali välillä $[4, \infty]$.

5. Tehtävän ratkaisussa on mahdollista käyttää apuna tehtävän 1 koodia.

```
1  import math
2
3  def f1(x):
4      #palauttaa eksponenttifunktion integraalifunktion
5      return math.e**x
6
7  def f2(x):
8      #palauttaa polynomin integraalifunktion
9      return x**3/3+x
10
11 #ensimmäisen osavälin alaraja
12 a = -math.inf
13
14 #ensimmäisen osavälin yläraja JA
15 #toisen osavälin alaraja
16 b = 0
17
18 #välin yläraja
19 c = 1
20
21 A1 = f1(b)-f1(a)
22 A2 = f2(c)-f2(b)
23
24 print(A1+A2)
```

6. Pilko-aliohjelmaan liittyvän vuokaavion mukaisesti voidaan muodostaa seuraava koodi.

```
1  def pilko(merkkijono):
2      lista = merkkijono.split('+')
3      return lista
4
5  f = "3x^3+2x^2+2x^1"
6  lista = pilko(f)
7  print("Lista funktion", f, "termeistä on", lista)
```

7. Termin derivaattaan liittyvän vuokaavion mukaisesti voidaan muodostaa seuraava koodi.

```
1  def derivaatta(termi, x):
2      #termistä saatu kerroin on aluksi merkkijono
3      kerroin = ""
4      i = 0
5      #termistä halutaan tarkastella sitä osaa, joka
6      #on ennen muuttujaa x
7      while(i<len(termi) and termi[i] != 'x'):
8          kerroin = kerroin + termi[i]
9          i = i + 1
10     #myös eksponentti saadaan aluksi merkkijonona
11     eksponentti = ""
12     #hypätään termin muuttujan x ja ^-merkin ylitse
13     j = i + 2
14     while(j < len(termi)):
15         eksponentti = eksponentti + termi[j]
16         j = j + 1
17     #muutetaan eksponentti ja kerroin kokonaisluvuiksi
18     e = int(eksponentti)
19     k = int(kerroin)
20     derivaatta = k * e * x**(e-1)
21     return derivaatta
22
23
24 termi = "3x^2"
25 x = 2
26 #kutsutaan funktiota derivaatta ja tallennetaan sen
27 #palauttama arvo
28 derivaatta = derivaatta(termi, x)
29 print("Termin", termi, "derivaatta pisteessä x =", x,
30       "on", derivaatta)
```

8. Tässä on yksi tapa ratkaista funktion derivointitehtävä.

```
def derivaatta(termi, x):
    #termistä saatu kerroin on aluksi merkkijono
    kerroin = ""
    i = 0
    #termistä halutaan tarkastella sitä osaa, joka
    #on ennen muuttujaa x
    while(i<len(termi) and termi[i] != 'x'):
        kerroin = kerroin + termi[i]
        i = i + 1
    #myös eksponentti saadaan aluksi merkkijonona
    eksponentti = ""
    #hypätään termin muuttujan x ja ^-merkin ylitse
    j = i + 2
    while(j < len(termi)):
        eksponentti = eksponentti + termi[j]
        j = j + 1
    #muutetaan eksponentti ja kerroin kokonaisluvuiksi
    e = int(eksponentti)
    k = int(kerroin)
    derivaatta = k * e * x**(e-1)
    return derivaatta

def pilko(merkkijono):
    lista = merkkijono.split('+')
    return lista

funktio = "3x^2+2x^2+2x^1"
x = 2
lista = pilko(funktio)
i = 0
vastaus = 0
#derivoidaan jokainen funktion termi derivaatta-aliohjelman
#avulla
while(i < len(lista)):
    vastaus = (vastaus + derivaatta(lista[i], x))
    i = i + 1
print("Funktion", funktio, "derivaatta pisteessä x =", x,
      "on", vastaus)
```

BONUSTEHTÄVÄ

(a) Muokkaa tekemääsi ohjelmakoodia siten, että käyttäjän on mahdollista antaa ohjelmalle oma funktio ja piste x.

Ohjelmakoodissa muuttuvat vain rivit 27 ja 28. Huomaa, että ohjelmalle ei voi antaa funktiota, jossa on vähennyslaskua.

```
funktio = input("Anna derivoitava funktio: ")
x = int(input("Anna piste, jossa funktio derivoidaan: "))
```

(b) Mieti, mitä ohjelmakoodissa tulisi ottaa huomioon, jotta käyttäjän olisi mahdollista syöttää funktio, jossa on myös termien vähennyslaskua.

Ohjelmakoodiin on mahdollista syöttää funktio, jossa negatiivinen termi lasketaan yhteen muiden termien kanssa. Esimerkiksi funktio $x^2 - 2x - 4$ voidaan antaa ohjelmalle muodossa `x^2+-2x^1+-4x^0`. Jos ohjelmakoodiin haluttaisiin kuitenkin syöttää myös vähennyslaskua, tulisi miettiä ainakin seuraavia asioita:

- `pilko`-aliohjelman tulisi katkaista merkkijono sekä plus- että miinusmerkkien kohdalta.
- Laskettaessa vastausta ohjelmakoodin tulisi muistaa, onko termi ollut positiivinen vai negatiivinen, jotta tiedetään, lisätäänkö termin derivaatta vastaukseen vai vähennetäänkö se.

9. Integraalifunktion laskentatehtävän ratkaisuun voi käyttää tehtävän 8 ohjelmakoodia pienillä muutoksilla. Ainut toiminnallinen muutos tulee riveille 20–21. Selkeyden vuoksi ohjelmaan tulee muuttaa myös aliohjelman nimi ja tulostettava sisältö.

Esimerkkirivit 20–21:

```
integraali = k * 1/(e+1) * x**(e+1)
return integraali
```

10. Alla olevassa koodissa on pelkkä pääohjelma määrätyn integraalin laskevaan tehtävään. Lisää koodin alkuun tässä tarvittavat aliohjelmat pilko (tehtävästä 6) ja integraali (muuta tehtävän 8 derivaatta-aliohjelman nimeksi integraali ja vaihda siihen tehtävän 9 ratkaisun yhteydessä esitellyt rivit).

```
1  funktio = input("Anna integroitava funktio: ")
2  x = int(input("Anna integraalin alaraja: "))
3  y = int(input("Anna integraalin yläraja: "))
4  lista = pilko(funktio)
5
6  #integroidaan jokainen funktion termi integraalifunktion
7  #avulla pisteessä x ja pisteessä y
8  i = 0
9  vastaus1 = 0
10 while(i < len(lista)):
11     vastaus1 = (vastaus1 + integraali(lista[i], x))
12     i = i + 1
13
14 j = 0
15 vastaus2 = 0
16 while(j < len(lista)):
17     vastaus2 = (vastaus2 + integraali(lista[j], y))
18     j = j + 1
19
20 print("Integraali annetulla välillä on", vastaus2-vastaus1)
```

11. Lisää itse tähän odotusarvon laskevan koodin alkuun aliohjelmat pilko ja integraali (samalla tavoin kuin tehtävän 10 ratkaisussa).

    Riveillä 1–27 kohdan (c) vastaus. Riveillä 34–41 kohdan (d) vastaus. Riveille 47, 48, 53 ja 54 muutettu muuttujan `lista` tilalle muuttuja `kerrotutTermit` kohdan (e) mukaisesti.

```
1  #funktio palauttaa termin siten, että x:n eksponenttia on
2  #kasvatettu yhdellä
3  def odotusarvo(termi):
4      i = 0
5      #kopioidaan termin alkuosa ennen ^-merkkiä uusiTermi-
6      #muuttujaan
7      uusiTermi = ""
8      while(termi[i] != '^'):
9          uusiTermi = uusiTermi + termi[i]
10         i = i + 1
11     #lisätään ^-merkki uusiTermi-muuttujaan ja kasvatetaan
12     #muuttujan i arvoa yhdellä
13     uusiTermi = uusiTermi + termi[i]
14     i = i + 1
15     #kopioidaan eksponentti muuttujaan eksponentti
16     eksponentti = ""
17     while(i < len(termi)):
18         eksponentti = eksponentti + termi[i]
19         i = i + 1
20     #muutetaan eksponentti lukuarvoksi
21     eksponenttiLukuna = int(eksponentti)
22     #kasvatetaan eksponentin arvoa yhdellä
23     eksponenttiLukuna = eksponenttiLukuna + 1
24     #lisätään eksponentti merkkijonona uusiTermi-muuttujan
25     #loppuun
26     uusiTermi = uusiTermi + str(eksponenttiLukuna)
27     return uusiTermi
28
29 funktio = input("Anna tiheysfunktio: ")
30 x = int(input("Anna integraalin alaraja: "))
31 y = int(input("Anna integraalin yläraja: "))
32 lista = pilko(funktio)
33
34 #kutsutaan odotusarvo-aliohjelmaa jokaisella funktion
```

```
35 #termillä ja lisätään funktion palauttama termi uuteen
36 #listaan kerrotutTermit
37 k = 0
38 kerrotutTermit = []
39 while(k < len(lista)):
40     kerrotutTermit.append(odotusarvo(lista[k]))
41     k = k + 1
42
43 #integroidaan jokainen funktion termi integraalifunktion
44 #avulla pisteessä x ja pisteessä y
45 i = 0
46 vastaus1 = 0
47 while(i < len(kerrotutTermit)):
48     vastaus1 = (vastaus1 + integraali(kerrotutTermit[i], x))
49     i = i + 1
50
51 j = 0
52 vastaus2 = 0
53 while(j < len(kerrotutTermit)):
54     vastaus2 = (vastaus2 + integraali(kerrotutTermit[j], y))
55     j = j + 1
56
57 print("Satunnaismuuttujan X odotusarvo on",
58       vastaus2-vastaus1)
```

Odotusarvo on $E(X) \approx 2,6667$.

12. Tässä aliohjelmaa toisesta tiedostosta kutsuvassa koodissa on muokattu tehtävän 11 vastausta. Aliohjelmat on siirretty tiedostoon aliohjelmaTiedosto.py, jota kutsutaan pääohjelman alussa. Ohjelmakoodeja on lyhennetty tilan säästämisen vuoksi.

    Pääohjelma:

```
1  from aliohjelmaTiedosto import *
2
3  funktio = input("Anna tiheysfunktio: ")
4  x = int(input("Anna integraalin alaraja: "))
5  y = int(input("Anna integraalin yläraja: "))
6
7  lista = pilko(funktio)
8  .
9  .
10 .
11
12 print("Satunnaismuuttujan X odotusarvo on",
13       vastaus2-vastaus1)
```

    Aliohjelmat:

```
1  def integraali(termi, x):
2      .
3      .
4      .
5      return integraali
6
7  def pilko(merkkijono):
8      lista = merkkijono.split('+')
9      return lista
10
11 def odotusarvo(termi):
12     .
13     .
14     .
15     return uusiTermi
```

13. Keskihajonta

    (a) Kaavaa käytettäessä tehdään seuraavat laskutoimitukset: Odotusarvon laskeminen, binomin neliön ottaminen, kahden funktion tulo, määrätyn integraalin laskeminen ja neliöjuuren ottaminen.

    (b) Valmiit aliohjelmat ratkaisevat odotusarvon ja määrätyn integraalin. Neliöjuureen on mahdollista käyttää math-kirjaston sqrt-funktiota.

    (c) Kirjoita <???> tilalle sen tiedoston nimi, jossa halutut aliohjelmat ovat. Muista, että tiedoston tulee olla tallennettu samaan kansioon kuin pääohjelmatiedoston.

    (d) Muista ottaa käyttöön kaikki tarvittavat tiedostot import-komennolla. Tehtävän 11 pääohjelman loppuun voidaan lisätä seuraavat rivit, jotka suorittavat halutut toiminnot.

```
odotusarvo = vastaus2-vastaus1
kh = keskihajonta(odotusarvo, funktio, x, y)
print("Satunnaismuuttujan X keskihajonta on "
    "D(X)=", kh)
```

    (e) Tiheysfunktion $f(x) = -x^3 + 4x$ keskihajonta on $D(X) \approx 6,461$.

BONUSTEHTÄVÄ Tavallisesti muuttujat, jotka luodaan aliohjelman sisällä ovat määritelty vain kyseistä aliohjelmaa käytettäessä eikä niitä voida suoraan käyttää toisessa aliohjelmassa tai pääohjelmassa. Aliohjelmassa `kerroin` on määritelty global-muuttuja `i`, joka mahdollistaa kyseisen muuttujan käytön myös muissa aliohjelmissa ja pääohjelmassa. Esimerkiksi, jos `kerroin`-aliohjelman while-silmukka kasvattaa muuttujaa `i` kahdella (siis while-silmukan jälkeen `i = 2`) ja tämän jälkeen `eksponentti`-aliohjelmassa käytetään komentoja `global i` ja `i = i + 2`, niin nyt global-muuttuja `i` on saanut arvon 4.

Lisää tietoa voi etsiä internetistä hakusanoilla Python global variables.

LÄHTEET

[1] Hähkiöniemi, M., Juhala, S., Juutinen, P., Laitinen, A., Luoma-aho, E., Raittila, T. & Tikka, T. *Juuri 11 Algoritmit ja lukuteoria*, Kustannusyhtiö Otava, Keuruu, 2021. ISBN 978-951-1-36302-6.

[2] Juhala, S., Juutinen, P., Laitinen, A., Luoma-aho, E., Raittila, T. & Tikka, T. *Juuri 12 Analyysi ja jatkuva jakauma*, Kustannusyhtiö Otava, Keuruu, 2022. ISBN 978-951-1-36302-6.

[3] Järvenpää, J. *Python-ohjelmoinnin opas opettajalle*, Otavan Kirjapaino Oy, Keuruu 2019. ISBN 978-951-1-36302-6.

[4] W3schools. *Python - Global Variables*, https://www.w3schools.com/python/python_variables_global.asp

[5] Rossum, G., Warsaw B. & Coghlan A. *PEP 8 - Style Guide for Python Code* https://peps.python.org/pep-0008/ 2001.

Kansikuva: Pixabay https://pixabay.com/users/clker-free-vector-images-3736/
Vuokaaviot on suunniteltu ja toteutettu Canva-suunnitteluohjelmalla https://www.canva.com/fi_fi/.

LIITE B: ALKUKYSELY

Ennen MAA12-opintojaksoa suoritettava kysely

Kysely on tarkoitettu opiskelijoille, jotka haluavat tehdä opintojaksoon liittyviä Python-ohjelmointitehtäviä. Kysely kartoittaa lyhyesti opiskelijan esitietoja ja ohjelmoinnin herättämiä tunteita.

* Lomake tallentaa nimesi. Kirjoita nimesi.

1

Olen suorittanut MAA11-opintojakson?

- ◯ Kyllä
- ◯ Ei

2

Tarkastele oheista koodia. Mitä ohjelma tulostaa?

```
for i in range (1, 6):
    if i % 2 == 0:
        print (i)
```

3

Kerro lyhyesti, mitä eroa on komennoilla **for** ja **while**.

4

Järjestä ohjelmoinnin termit helpoimmasta vaikeimpaan (helpoin ylimmäksi).

- **for**-toistorakenne
- **while**-toistorakenne
- **if-else**–valintarakenne
- **lista** tietorakenteena
- **input()**-komento

5

Yleiset kysymykset opintojaksosta

	Täysin eri mieltä	Osittain eri mieltä	En osaa sanoa	Osittain samaa mieltä	Täysin samaa mieltä
Olen innoissani tulevasta opintojaksosta	◯	◯	◯	◯	◯
Olen huolissani, ymmärränkö opintojakson materiaalia	◯	◯	◯	◯	◯
Uskon, että opin paljon uutta opintojakson aikana	◯	◯	◯	◯	◯
Minua jännittää, kun ajattelen opintojakson sisältöä	◯	◯	◯	◯	◯
Olen varma, että tulen saavuttamaan opintojakson oppimistavoitteet	◯	◯	◯	◯	◯

6

Kysymykset koskien ohjelmointia opintojaksolla

	Täysin eri mieltä	Osittain eri mieltä	En osaa sanoa / kysymys ei koske minua	Osittain samaa mieltä	Täysin samaa mieltä
Ohjelmoinnin opiskelu kiinnostaa minua	◯	◯	◯	◯	◯
Olen huolissani ohjelmointitehtävien tason haastavuudesta	◯	◯	◯	◯	◯
MAA11-opintojakso motivoi minua opiskelemaan ohjelmointia lisää	◯	◯	◯	◯	◯
Ohjelmointitehtävien ajatteleminen pelottaa minua	◯	◯	◯	◯	◯
Uskon selviytyväni helposti tulevista ohjelmointitehtävistä	◯	◯	◯	◯	◯

LIITE C: LOPPUKYSELY

MAA12-opintojakson jälkeen suoritettava kysely

Kysely on tarkoitettu opiskelijoille, jotka tekivät MAA12-opintojaksoon liitettyjä ohjelmointitehtäviä (yhden tai useampia). Kyselyn tarkoituksena on kerätä palautetta ohjelmointimateriaalista, jotta sitä voidaan jatkossa kehittää. Kyselyyn vastaaminen ei vaikuta kurssin arvosanaan.

* Pakollinen

* Lomake tallentaa nimesi. Kirjoita nimesi.

1. Merkitse ne tehtävät, jotka ratkaisit kurssin ohjelmointimateriaalista. *

   Tähän kysymykseen vastaaminen ei vaikuta kurssisuoritukseesi tai arvosanaasi, vaan kysytään pelkästään materiaalin kehittämistä varten.
   Riittää, että olet yrittänyt tehdä tehtävää, koodin ei tarvitse olla valmis tai toimiva.

   - [ ] 1. Puuttuvat arvot
   - [ ] 2. Osamäärän raja-arvo
   - [ ] 3. Funktion raja-arvo
   - [ ] 4. Kaavarivit solmussa
   - [ ] 5. Funktion ja x-akselin välinen pinta-ala
   - [ ] 6. Pilko-aliohjelma
   - [ ] 7. Termin derivaatta
   - [ ] 8. Funktion derivaatta
   - [ ] BONUSTEHTÄVÄ 1
   - [ ] 9. Integraalifunktion laskeminen
   - [ ] 10. Määrätty integraali
   - [ ] 11. Odotusarvo
   - [ ] 12. Kaksi tiedostoa
   - [ ] 13. Keskihajonta
   - [ ] BONUSTEHTÄVÄ 2

2. Jos et suorittanut kaikkia kurssimateriaalin tehtäviä, kerro lyhyesti miksi et. *

3. Valitse jokaista tehtävää parhaiten kuvaava vaihtoehto. *

	Aivan liian haastava	Hieman liian haastava	Sopivan haastava	Hieman liian helppo	Aivan liian helppo
Puuttuvat arvot	◯	◯	◯	◯	◯
Osamäärän raja-arvo	◯	◯	◯	◯	◯
Funktion raja-arvo	◯	◯	◯	◯	◯
Kaavarivit solmussa	◯	◯	◯	◯	◯
Funktion ja x-akselin välinen pinta-ala	◯	◯	◯	◯	◯
Pilko-aliohjelma	◯	◯	◯	◯	◯
Termin derivaatta	◯	◯	◯	◯	◯
Funktion derivaatta	◯	◯	◯	◯	◯
BONUSTEHTÄVÄ 1	◯	◯	◯	◯	◯
Integraalifunktion laskeminen	◯	◯	◯	◯	◯
Määrätty integraali	◯	◯	◯	◯	◯
Odotusarvo	◯	◯	◯	◯	◯
Kaksi tiedostoa	◯	◯	◯	◯	◯
Keskihajonta	◯	◯	◯	◯	◯
BONUSTEHTÄVÄ 2	◯	◯	◯	◯	◯

4. Minkä ohjelmointiin liittyvän asian koet ymmärtäneesi erityisen hyvin opintojakson aikana? *

5. Mikä ohjelmointiin liittyvä asia jäi epäselväksi? *

6. Arvioi seuraavia väitteitä kuluneen opintojakson ohjelmointimateriaalista *

	Vaihtoehto 1	Vaihtoehto 2	Vaihtoehto 3	Vaihtoehto 4	Vaihtoehto 5
Uudet komennot ja tietorakenteet oli selitetty selkeästi	◯	◯	◯	◯	◯
Tehtävänannot olivat helposti ymmärrettäviä	◯	◯	◯	◯	◯
Keksin nopeasti, millaisella algoritmilla tehtävän voi toteuttaa	◯	◯	◯	◯	◯
Olisin kaivannut enemmän esimerkkejä	◯	◯	◯	◯	◯
Ohjelmointitehtävien tekeminen auttoi minua ymmärtämään kurssin sisältöä paremmin	◯	◯	◯	◯	◯

7. Valitse jokaiseen väittämään parhaiten tunteitasi kuvaava vaihtoehto *

	Täysin eri mieltä	Osittain eri mieltä	En osaa sanoa	Osittain samaa mieltä	Täysin samaa mieltä
Olen iloinen, sillä ymmärsin opintojakson materiaalin	◯	◯	◯	◯	◯
Minua ärsyttää, sillä muut ymmärsivät tehtävät paremmin kuin minä	◯	◯	◯	◯	◯
Koen voivani olla ylpeä siitä, mitä osaan	◯	◯	◯	◯	◯
Odotan innolla uusien ohjelmointitaitojen oppimista	◯	◯	◯	◯	◯
Haluan kertoa kavereilleni tästä kurssista	◯	◯	◯	◯	◯

8. Vapaa sana (esim. päällimmäinen tunne kurssista, miten sinä kehittäisit kurssin sisältöä, pitäisikö ohjelmoinnin opetusta olla enemmän jne.). *