Roope Ikonen

PYTHON PERUSTEET

Liiketalous
2021

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Roope Ikonen
Opinnäytetyön nimi	Python Perusteet
Vuosi	2021
Kieli	suomi
Sivumäärä	32
Ohjaaja	Klaus Salonen

Tämän opinnäytetyön tarkoitus on antaa lukijalle tarvittavat työkalut, joilla pääsee Python-ohjelmoinnissa alkuun. Työn on tarkoitus auttaa uutta ohjelmoijaa oppimaan Python-ohjelmoinnin perusteita muuttujista funktioihin.

Opittavat asiat on pyritty tekemään helposti luettavalla teorialla. Kuvia on pyritty käyttämään mahdollisimman paljon auttamaan aiheiden sisäistämiseen ja ohjelman koodin tuottamiseen. Työhön on myös luotu tehtäviä, joiden on tarkoitus auttaa tietyn asia alueen oppimisessa.

Lopputuloksena opinnäytetyöhön saatiin teorian lisäksi neljä tehtävää. Tehtävinä on kolme pienempää tehtävää ja yksi lopputehtävä. Lopputehtävä pyrkii hyödyntämään lähes kaikkea opinnäytetyössä opittua materiaalia.

Avainsanat	Python, ohjelmointi, perusteet, oppimateriaali

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Informational Technology

ABSTRACT

Author	Roope Ikonen
Title	Python basics
Year	2021
Language	Finnish
Pages	32
Name of Supervisor	Klaus Salonen

The aim of the thesis was to give a starting programmer the basic tools needed to start programming with Python. The thesis covers topics from variables to basic functions.

The material was produced with the idea of making the text as easy to follow as possible for beginners. To support the text the material has a lot of pictures of code to the user with visualizing how the program and material should work. Also, to help with learning the material there are assignments that aim at internalizing the material.

The result in addition to the text was four tasks, three smaller ones and one bigger for the final assignment. The final assignment aims to use the knowledge from almost all the topics learned before.

Keywords Python, basics, learning material, programming

SISÄLLYS

KUVIO- JA TAULUKKOLUETTELO

1 JOHDANTO

Opinnäytetyön tavoitteena on perehtyä Python ohjelmointikielen perusteisiin ja saada selkeät suomenkieliset ohjeet henkilöille, jotka eivät mahdollisesti ole taitavia englannin kielessä. Suurimmat osat ohjeista netissä ovat kuitenkin ovat englanniksi. Tarkoituksena on aloittaa käyttöönotosta ja käydä aihealueita läpi, jotka valmistavat lukijan tekemään lopputehtävän, joka käyttää opittuja aiheita. Teksti on pyritty pitämään mahdollisimman tiiviinä välttäen liikaa teoriaa mutta kumminkin yritetty sisällyttää kaikki aiheeseen liittyvät tärkeät asiat. Teksti on myös yritetty pitää mahdollisimman aloittelijaystävällisenä.

Opinnäytetyön on tarkoitus ohjeistaa lukijaa käyttäen teoriaa ja kuvia koodista. Lukijalle oppimisen tukemiseksi joistain kappaleista löytyy tehtäviä, jotka valmistavat lukijaa lopputehtävän tekemiseen. Opinnäytetyö on toiminnallinen ja toteutetaan projektina auttamaan Python kielen opiskelussa tai miksi ei vaikka opettamisessakin.

Opinnäytetyöni päätarkoituksena on tehdä suomenkieliset ohjeet Python-ohjelmointi kielen perusteista. Suurin osa ohjeista, joita netistä löytyy, on englannin kielellä. Opinnäytetyön on myös tarkoitus opettaa itselle paremmin Python-ohjelmointi kieltä. Aikaisemmin olen tehnyt vähän Python ohjelmointia mutta se oli kokonaan verkossa, joten en tiennyt kunnolla, miten se toimii käytännössä. Ohjelmoinnista muilla kielillä minulla oli kokemusta ennen työn aloittamista.

Tavoitteena on saada teorian, kuvien ja tehtävien avulla lukija ymmärtämään Python-ohjelmointi kielen perusteet ja kokeilemaan itse tehtävien avulla, kuinka kieli toimii. Tehtävät on suunniteltu auttamaan sisäistämään kappaleessa käsitelty teoria.

2 PYTHON ALUSTAVAA TEORIAA

Python on monipuolinen ohjelmointikieli, jota voidaan käyttää esimerkiksi verkko-ohjelmointiin, tekoälyn tekoon, pelien tekoon, koneoppimiseen ja monen muun asian tekemiseen. Python on hyvä ensimmäiseksi ohjelmointikieleksi, koska se on helppo oppia ja käyttää. Tämä on suurimmaksi osaksi helpon syntaksin takia. (McKeown 2019). Tämä tarkoittaa sitä, että tarvitsee kirjoittaa vähemmän koodia kuin muilla kielillä ja että syntaksi muistuttaa enemmän normaalia englannin kieltä.

Pythonilla on myös huonoja puolia kuten kaikilla muillakin ohjelmointi kielillä. Helpon syntaksin takia voi olla vaikeampi oppia muita kieliä, jos on kerennyt tottumaan helppoon syntaksiin, jossa ei tarvitse käyttää esimerkiksi aaltosulkeita. Python toimii myös hitaammin kuin muut kielet kuten vaikka Java. Myös koska, Python vaatii paljon muistia ja suoritinaikaa sitä ei suositella mobiili sovelluksien tekoon mobiilisovellukset, kun yleensä tehdään käyttämään mahdollisimman vähän resursseja (Malik 2019).

3 PYTHON ALOITUS

Tässä luvussa käydään asioita läpi, jotka auttavat aloittamaan Pythonin käyttöä kuten käyttöönotto, print-komento ja muuttujat. Tarkoitus käydä siis ihan perusteet läpi aika lyhyesti.

3.1 Käyttöönotto

Ennen kun pääset kirjoittamaan koodia, joudut asentamaan Pythonin tietokoneellesi, mikäli se ei ole jo tietokoneellasi joissain tietokone malleissa saattaa olla valmiina. Mikäli Pythonia ei löydy tietokoneeltasi mene osoitteeseen https://www.python.org/ ja valitse sivuilta downloads kohta. Downloads kohdassa sivut ehdottavat heti ensimmäiseni uusinta versiota, klikkaa sitä niin asennus tiedosto latautuu tietokoneellesi. Seuraavaksi valitset mihin haluat asentaa tiedoston ja mitä osia haluat asentaa siitä. Itse asentaessa seurasin vain suosituksia.

Seuraavaksi mikäli haluat voit asentaa esimerkiksi Visual Studio Coden ja sieltä Extensions kohdasta Python lisäosan Visual Studioon. Itse käytin tätä hyödyksi tässä työssä. Se tekee koodista helpomman lukea, kuin jos käyttäisi pelkkää Python terminaalia koska siellä koodi ja tulostukset tulevat päällekkäin, kun taas Visual Studio Codea käyttämällä saat kirjoitettua koodin eri osaan ohjelmaa kuin minne terminaali tulostaa näin pitäen koodin ja tulostuksen erillään ja helpompana lukea.

3.2 Print funktio

Print-funktio tulostaa tietyn viestin näytölle tai mihin ulostulo tuleekaan. Viesti voi olla string-muodossa tai jos se on jossain muussa muodossa, se muuttuu string muotoon, kun tulostuu näytölle (Python 2021, A.).

Alla olevassa Kuvan 1 esimerkissä on näytetty, kuinka print-komento toimii aluksi, lause on laitettu muuttujaan josta toinen print-komento tulostaa konsoliin tekstin

ja toisen print-komennon sisään on laitettu pelkkä teksti tulostettavaksi. Esimerkkikuvassa on eri heittomerkit laitettu näyttämään, että sillä ei ole merkitystä kumpia käytät Python ohjelmoinnissa koska molemmat ajavat saman asian.

```
i= 'Hello world!'

print("Terve maailma!")
print(i)
```

Kuva 1. Print esimerkki

Kuvan 1 esimerkki tulostaa konsoliin ylhäältä alaspäin lauseet ”Terve maailma!” ja ”Hello world!” päällekkäin.

3.3 Python muuttujat

Muuttujan voisi helposti selittää niin, että annat tietylle objektille nimen. Tätä objektia sitten kutsutaan käyttämällä sille annettua nimeä. Muuttujan voi nimetä kuinka haluaa isoilla ja pienillä kirjaimilla tai numeroilla. Ainut rajoitus on, että nimi ei voi alkaa numerolla.

Monissa ohjelmointikielissä muuttujille pitää aluksi määritellä jokin tietotyyppi ja se joutuu pysymään siinä tietotyypissä niin kauan kun sitä käytetään. Pythonissa muuttujat eivät ole rajoitettu tiettyyn tietotyyppiin samalle muuttujalle voi myöhemmin määrätä erityyppisen arvon (Sturtz 2021).

```
x = 3
y = "kolme"
z = 3.0
```

Kuva 2. Muuttujatyyppejä

Kuvassa 2 näkyy kolme eri muuttuja tyyppiä int, str ja float. Muuttujan voi kutsua käyttämällä tässä tapauksessa, vaikka ”print(x)” jolloin konsoli tulostaa x muuttujan sisällön. Muuttujan sisällön voi myös laittaa sulkujen sisälle, mikäli haluaa mutta ei ole pakollista Pythonissa.

Jos et ole varma minkä tyyppinen muuttujasi on ja haluat saada sen selville tiettyyn tarkoitukseen voit käyttää type-komentoa.

```
print(type(x))
print(type(y))
print(type(z))
```

Kuva 3. Type-komento

Kuvan 3 esimerkissä type-komentoa on käytetty samoihin muuttujiin kuin aikaisemmassa esimerkkikuvassa.

```
<class 'int'>
<class 'str'>
<class 'float'>
```

Kuva 4. Typen konsoli tulostus

Type-komentoa käyttämällä aikaisemman kuvan mukaisesti on tulostus konsoliin tämä. Heittomerkkien sisällä on muuttujan tyyppi.

3.4 Python matemaattiset operaattorit

Pythonissa kuten muissakin ohjelmointikielissä käytetään perusmatemaattisia operaattoreita kuten esimerkiksi summaus ja jako. Operaattoreita käytetään esimerkiksi kahden arvon lisäämiseen toisiinsa tai muuttujaan arvon lisäämiseen.

```
x = 5
y = 5

i = x * y
```

Kuva 5. Muuttujien keskenään kertominen

Muuttujien x ja y arvot kerrotaan keskenään ja niiden tulos annetaan muuttujalle i. Muuttujan sijasta voisi myös käyttää pelkkiä numeroja ja lisätä ne muuttujaan i.

Taulukko 1. Matemaattiset operaattorit

x + y	Arvojen lisääminen toisiinsa
x - y	Arvojen vähennys toisistaan
x * y	Arvojen tulo
x / y	Arvojen jako
x % y	Arvojen jakojäännös
x // y	Arvojen tasajako
x ** y	Arvojen potenssi

4 IF- LAUSEET

If-lause on yksinkertaisin päätöksentekolause. If-lausetta käytetään päättämään, suoritetaanko pala koodia vai ei. Jos If-lauseen sisältö on tosi, se suoritetaan tai jos se on epätosi, se osa koodista ohitetaan. (Agarwal 2021).

```
x = 6
y = 5
if (x > y):
    print("x on suurempi kuin y")
```

Kuva 6. IF-lause esimerkki

Kuvassa 6 näkyy esimerkki, jossa jos muuttuja x on suurempi kuin y, teksti tulostuu. Koodin porrastus on tärkeää koska, jos print-komento alkaisi suoraan if-ehdon alapuolelta tulisi virheilmoitus.

Taulukko 2. If-lause vertailu operaattorit

X == Y	Yhtä suuri kuin
X != Y	Erisuuri kuin
X < Y	Pienempi kuin
X <= Y	Pienempi tai yhtä suuri kuin
X > Y	Suurempi kuin
X >= Y	Suurempi tai yhtä suuri kuin

4.1 Elif-lause

Elif-lause tulee käyttöön, jos haluat lisätä if-lauseeseen ehdon, että mikäli aiempi lause ei toteudu toteuta tämä.

```
x = 6
y = 7
if x > y:
    print("x on suurempi kuin y")
elif x < y:
    print("x on pienempi kuin y")
```

Kuva 7. Elif esimerkki

Esimerkki katsoo kumpi lause käy toteen ja tulostaa vastauksen sen mukaan. Tässä esimerkissä tulostus on ”x on pienempi kuin y.”

4.2 Else-lause

Else on ehto, joka toteutuu, jos mikään aikaisemmista ehdoista ei käy toteen. Eli jos if- tai elif-lauseiden ehdot eivät käy toteen ohjelma tulostaa elsen.

```
x = 6
y = 6
if x > y:
    print("x on suurempi kuin y")
elif x < y:
    print("x on pienempi kuin y")
else:
    print("x ja y ovat yhtä suuria")
```

Kuva 8. Else esimerkki

Esimerkin koodi tarkistaa ensin toteutuuko if- lauseen ehdot sitten elif-lauseen ehdot ja kun kummatkaan eivät toteudu se tulostaa elsen sisällön.

4.3 Useamman ehdon if-lause

Python ohjelmoinnissa on kaksi pääoperaattoria, jotka voidaan yhdistää if-lauseisiin and ja or. And toteutuu jos molemmat ehdot ovat lauseessa tosi. Or toteutuu, jos jompikumpi ehdoista on tosi. (W3Schools 2021, B)

```
x = 6
y = 5
z = 7

if x > y and y < z:
    print("molemmat ehdot ovat tosi!")

if x == y or z >= y:
    print ("vähintään toinen ehdoista on tosi!")
```

Kuva 9. And ja or esimerkki.

4.4 If harjoitustehtävä

Harjoitustehtäväksi tee ohjelma, joka vertaa muuttujia keskenään. Ohjelman if-lause tulostaa konsoliin tosi, jos muuttuja on pienempi tai yhtä suuri kuin viisi jos muuttuja ei ole tosi if-lauseen ehtojen mukaan niin tulosta konsoliin epätosi käyttäen else-lausetta.

4.5 If tehtävän esimerkkivastaus

Ensin ohjelmaan on määritelty muuttaja, jonka arvoksi on annettu neljä. Sitten tehty if-lause mikä tarkistaa onko muuttujan arvo pienempi tai yhtä suuri kuin viisi. Jos ehto on tosi eli muuttuja on pienempi tai yhtä suuri niin tulostuu ”Tosi” jos muuttuja on jotain muuta kuin pienempi tai yhtä suuri niin tulostuu ”Epätosi”.

```
x = 4

if x <= 5:
    print("Tosi")
else:
    print("Epätosi")
```

Kuva 10. If tehtävän esimerkkivastaus

5 PYTHON SILMUKAT

Pythonin kaksi perussilmukkaa ovat while- ja for-silmukka. Silmukkoja käytetään yleensä koodiin mikä, halutaan toistaa tietty määrä tai toistuu siihen asti, kunnes sille annettu ehto täyttyy (Python 2021, A). Molemmat silmukat käyttävät myös continue- ja break-komentoja. Break-komentoa käytetään poistumaan silmukasta ja continue-komentoa käytetään nykyisen lohkon ohitukseen. Komennot sijoitetaan silmukan sisälle. (Learnpython. 2021)

5.1 For-silmukka

For-silmukkaa voidaan käyttää esimerkiksi listan, string-tyyppisen muuttujan ja numeroiden toistamiseen. For-silmukkaa voidaan käyttää numerojen toistamiseen range-funktiolla. (Learnpython. 2021)

```
lista = ["yksi", "kaksi", "kolme"]
for x in lista:
    print(x)
```

Kuva 11. For-silmukan lista esimerkki

Kuvan 11 esimerkki ohjelma tulostaa konsoliin allekkain listasta heittomerkeissä olevat sanat. Koodiin voisi lisätä print-komennon alapuolelle if-lauseen mikä, jos muuttuja on yhtä suuri kuin vaikka kaksi käyttää break-komentoa ja lopettaa koodin siihen.

```
for x in range(7):
    print(x)
```

Kuva 12. For-silmukka range() esimerkki

Kuvan 12 ohjelma tulostaa numerot nollasta kuuteen koska range-komennon sulkujen sisällä olevaa numeroa ei tulosteta. Jos siis haluaisi tulostaa numerot yhdeksään asti laitettaisiin sulkujen sisälle numero kymmenen.

```
for x in range(1, 7):
    print(x)
```

Kuva 13. For-silmukka range() esimerkki 2

Tämä toimii samalla tavalla kuin aikaisempi esimerkki paitsi tulostus on yhdestä kuuteen. Alkaa siis ensimmäisestä numerosta ja tulostaa toiseen numeroon asti.

5.2 While-silmukka

While-silmukkaa käytetään enimmäkseen numeroihin mutta idea on muuten samalainen kuin for-silmukassa. While-silmukka toimii niin kauan, kun sen ehto on tosi. While-silmukkaa käytetään yleensä, jos ei olla varmoja montako kertaa silmukka halutaan toistaa. (Programiz 2021, A)

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Kuva 14. While-silmukka esimerkki

Kuvan 14 ohjelma toistaa ja tulostaa muuttujan niin kauan kunnes sille annettu ehto täyttyy. Eli ohjelma katsoo, onko muuttuja pienempi kuin kuusi. Jos luku on pienempi kuin kuusi tulostuu muuttujan sisältö ja sitten alin rivi lisää muuttujaan summaan yhden (+=) niin kauan kunnes muuttaja ei ole enää pienempi kuin kuusi.

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("muttuja i ei ole pienempi kuin 6")
```

Kuva 15. Else ja while-silmukka esimerkki

Kuvan 15 esimerkissä on muuten sama ohjelma kuin aikaisemmassa esimerkissä, mutta tähän on lisätty else-lause. Tässä ohjelmassa, kun muuttuja ei ole enää pienempi kuin kuusi tulostuu elsen sisällä oleva sisältö.

5.3 Silmukka tehtävä

Tee while-silmukka, joka lisää yhden numeron muuttujaan, kunnes muuttuja on suurempi kuin seitsemän mutta ohittaa tulostuksessa luvun neljä. Tässä tehtävässä tarvitsee käyttää if-lausetta silmukan sisällä ja continue-komentoa.

5.4 Silmukka tehtävän esimerkkivastaus

Tässä ohjelmassa tehtiin while-silmukka, joka toistuu niin kauan, kunnes muuttuja on yhtä suuri kuin seitsemän mutta numero neljä ohitetaan tulostuksessa. Aluksi tehdään muuttuja, jonka arvo on nolla. Seuraavaksi tehdään silmukka, joka lisää muuttujaan joka kierroksella yhden ja silmukan sisälle laitetaan if-lause, joka katsoo, onko muuttujan arvo neljä. Jos muuttujan arvo on neljä niin continue-komento ohittaa tulostuksen ja hyppää alkuun. Silmukka loppuu, kun muuttuja on suurempi kuin seitsemän. Konsoliin pitäisi tulostua siis yhdestä seitsemään mutta numeroa neljä ei tulostu.

```
x = 0
while x < 7:
    x += 1
    if x == 4:
        continue
    print(x)
```

Kuva 16. While tehtävän esimerkkivastaus.

6 PYTHON FUNKTIOT JA MODUULIT

Funktiot ovat paloja toistuvaa koodia, jotka määritellään ja jota voidaan kutsua ohjelmassa tarvittaessa. Funktiot helpottavat järjestämään ohjelmasi koodia ja funktiota voidaan kutsua tarvittaessa useampaan kertaan. Funktion teko aloitetaan def-käskyllä, jonka perään tulee funktion nimi, sulut ja puolipiste. Sitten sen alapuolelle aloitetaan kirjoittamaan koodia mitä funktio toistaa kutsuttaessa. Funktio kutsutaan sille määritellyllä nimellä. (Pythonbasics. 2021)

```
def ekaFunktio():
  print("Tämä on mun eka funktio")

ekaFunktio()
```

Kuva 17. Esimerkki funktio

Esimerkki kuvassa 17 on määritelty funktio aikaisemmin mainitusti ja se kutsutaan alapuolella. Tämän tulostus on siis teksti, joka on funktiossa print-komennon sisällä. Tässä kuten aikaisemmissakin esimerkeissä koodin porrastus on tärkeää, että Python osaa lukea sitä.

6.1 Funktion parametrit

Funktiosta saa paljon monipuolisemman lisäämällä sille parametrit. Parametrit ovat periaatteessa muuttujia, joihin voidaan hakea sisältö funktion ulkopuolelta, joka välittyy funktion sisälle. Parametrit nimetään funktion nimen jälkeen sulkujen sisälle. Parametrien kanssa pitää myös olla tarkka, jos antaa vaikka kahdet parametrit ja toinen ei saa tietoa siitä syntyy virhe konsoliin. (Programiz 2021, B)

```
def kokoNimi(eNimi, sNimi):
  print (sNimi + " " + eNimi)

kokoNimi("Roope", "Ikonen")
```

Kuva 18. Parametrit esimerkki

Funktio on laitettu pyytämään etu- ja sukunimen ja tulostamaan ne sukunimestä etunimeen. Tyhjät heittomerkit välissä toimivat välilyöntinä. Alhaalla kun funktiota kutsutaan, on sille annettu parametrit. Jos tällä esimerkillä se saisi vain esimerkiksi etunimen tulisi konsoliin virheilmoitus.

```
def kokoNimi(eNimi, sNimi = "tyhjä"):
```

Kuva 19. Parametrit esimerkki 2

Jos ei ole varmaa saako parametri arvon voi sille antaa oletusarvon tällä vältytään mahdolliselta virheilmoitukselta. Mikäli kaikille parametreille ei anneta oletusarvoa pitää nimetä funktiossa ensin se parametri, joka ei saa oletusarvoa muuten siitäkin tulee virheilmoitus.

6.2 Funktio return-lause

Return-lausetta käytetään, kun halutaan palauttaa arvo. Arvo palautetaan sinne missä funktiota on kutsuttu. (Programiz 2021, C)

```
def kolmeKertaa(luku):
  return 3 * luku

print (kolmeKertaa(3))
print (kolmeKertaa(4))
```

Kuva 20. Return-lause esimerkki

Kuvan 20 esimerkki toimii kuin kolme kertotaulu eli konsoliin tulostuu yhdeksän ja kaksitoista päällekkäin. Return lause siis palauttaa kolmella kerrottuna luvun minkä sen parametri sai ja se tulostuu.

6.3 Python moduulit

Moduuli on tiedosto, jonka sisällä on koodia mitä voidaan kutsua sen nimellä eri tiedostossa. Moduuleja käytetään isommissa ohjelmissa, jotta ne voidaan jakaa pienempiin ohjelmiin, joita on helpompi lukea ja organisoida. Moduulin sisäisiä koodeja voi myös hyvin uudelleen käyttää eri paikoissa. Tee funktio ja tallenna se Python tiedostoon (.py) ja nimeä Python tiedosto. Moduuli kutsutaan sen tiedostonimellä. Ohjelmassa missä haluat käyttää moduulia kutsut sen import-komennolla, jonka perään tulee nimi mitä olet käyttänyt. Muuttujat joita käytät itse tehdyssä moduulissa pitää erottaa pisteellä eli esimerkiksi tiedostonimi.muutujanimi() ja sulkujen sisälle tarvittavat arvot. (Programiz 2021, D)

Python kielessä on myös laaja kirjasto valmiita moduuleja. Jokaisella moduulilla on joku eri toiminto. Kirjoittamalla help ja sulut sulkujen sisään heittomerkit ja modules tulostuu konsoliin suuri lista, jossa on kaikki käytettävät moduulit. (Tutorialsteacher 2021, A)

Kuva 21. Moduuli kirjaston komento

Jos haluat ottaa selvää kirjastoon tulevista moduuleista voit korvata help sulkujen sisälle jonkun moduulin nimen kirjastosta mikä tulostui. Tulostukseen tulee silloin moduulin perustieto ja linkki, josta voit lukea tarkemmin tietystä moduulista.

```
help('getpass')
```

Kuva 22. Help komento tiettyyn moduuliin

```
Help on module getpass:

NAME
    getpass - Utilities to get a password and/or the current user name.

MODULE REFERENCE
    https://docs.python.org/3.9/library/getpass

    The following documentation is automatically generated from the Python
    source files.  It may be incomplete, incorrect or include features that
    are considered implementation detail and may vary between Python
    implementations.  When in doubt, consult the module reference at the
    location listed above.

-- More  --
```

Kuva 23. Kuvan 22 komennon tulostus

6.4 Random-moduuli

Mainitsen random-moduulin erikseen koska, sitä tullaan tarvitsemaan viimeisen tehtävän tekoon. Random-moduuli pitää kutsua kuin mikä muukin moduuli import-käskyllä. Random-moduulia voidaan käyttää esimerkiksi satunnaisten numerojen luontiin, listasta satunnaisesti elementin valintaan ja moneen muuhun. Esimerkiksi randint-komennolla saadaan määritellä kaksi numeroa, jonka väliltä ohjelma valitsee satunnaisen kokonaisluvun (Tutorialsteacher 2021, B). Tässäkin kohtaa niin kuin aikaisemmin mainittiin voi help-komennolla saada tietoa kaikista mahdollisista komennoista mitä moduulin mukana tulee.

```
import random

x = 0

x = random.randint(1, 20)
print(x)
```

Kuva 24. Random-moduuli ja randint

Esimerkkikuvan ohjelmaan on tuotu random-moduuli sitten tehty muuttuja, jonka arvo on nolla. Tämän jälkeen randint-komennolla annetaan muuttujalle uusi arvo, jonka jälkeen muuttujan arvo tulostetaan konsoliin. Tulostus on siis joka kerta luku väliltä yksi ja kaksikymmentä.

6.5 Random-moduuli tehtävä

Tee koodi missä annat kahdelle muuttujalle random-moduulilla ja randint-komennolla arvot. Sitten vertaat niitä keskenään if-lauseella ja teet tulostuksen mikä kertoo, kumpi on isompi.

6.6 Random-moduulin esimerkkivastaus

Tehtävän tarkoituksena oli kokeilla random-moduulia ja randint-komentoa. Sekä vertailla niitä käyttämällä if-lausetta Tämä ohjelma aloitetaan kutsumalla random-moduuli ja luodaan kaksi muuttujaa, jonka arvo on nolla. Muuttujille annetaan seuraavaksi satunnaisesti arvo randint-komennolla ja arvot tulostetaan konsoliin näkyvin. Sitten luodaan if-lause, joka vertaa kumpi muuttujista on suurempi ja jos molemmat ovat yhtä suuria tulostetaan ”tasapeli” konsoliin.

```python
import random

eka = 0
toka = 0

eka = random.randint(1, 10)
toka = random.randint(1, 10)
print(eka, toka)

if eka > toka:
    print("Eka on isompi kuin toka")
elif toka > eka:
    print("Toka on isompi kuin eka")
else:
    print("tasapeli")
```

Kuva 25. Random tehtävän esimerkkivastaus

7 LOPPUTEHTÄVÄ

Lopputehtävän tarkoituksena on käyttää kaikkia aiemmin opittuja osia ja muodostaa niistä yksi ohjelma. Lopputehtävässä tarvitset random-moduulia ja joudut sijoittamaan if-lauseen silmukan sisälle. Lopputehtävässä voit käyttää hyödyksi Random-moduuli kohdan tehtävää. Tehtävän voi toteuttaa monella eri tavalla, joten jos ohjelmasi koodi ei ole sama kuin esimerkki vastauksessa tärkeintä on lopputulostus.

7.1 Tehtävä

Lopputehtävänä on tehdä noppapeli ohjelma missä kaksi pelaajaa heittää noppaa toisiaan vastaan. Aina kun toinen pelaaja voittaa lisätään hänelle piste ja kun noppaa on heitetty useampaan kertaan, näytetään lopuksi, kumpi voitti useampaan kertaan ja tulostaa tilastot konsoliin.

7.2 Lopputehtävän esimerkkivastaus

Lopputehtävän tarkoitus oli luoda noppapeli käyttämällä aikaisemmin opittuja keinoja. Ensin tuodaan random-moduuli ohjelmaan ja luodaan kolme muuttujaa. Yksi muuttuja laskee, montako kertaa noppia on heitetty eli monesko kierros while-silmukassa on menossa ja toiset kaksi muuttujaa katsoo montako kertaa kumpikin, pelaaja on voittanut. Seuraavaksi aloitetaan while-silmukka, joka ensin valitsee randint-komennolla kahteen muuttujaan arvon yhden ja kuuden väliltä, jonka jälkeen ohjelma kertoo mitä kumpikin pelaaja heitti ja sitten if-lauseella verrataan niitä keskenään. Se kumpi pelaaja saa pisteen niin lisätään siihen muuttujaan yksi piste. Ohjelman loppuun tulostetaan, että peli on ohi ja sitten, kuinka monta pistettä kumpikin pelaaja sai.

```python
import random

kierros = 0
pelaajaPisteet = 0
tietokonePisteet = 0

while kierros < 2:

    pelaaja = random.randint(1, 6)
    tietokone = random.randint(1, 6)

    print("Pelaaja heitti: ", pelaaja)
    print("Tietokone heitti: ", tietokone)

    if pelaaja > tietokone:
        print("Sinä voitit")
        pelaajaPisteet = pelaajaPisteet + 1
    elif tietokone > pelaaja:
        print("Tietokone voitti")
        tietokonePisteet = tietokonePisteet + 1
    else:
        print("Tasapeli")
    kierros += 1

print("Peli ohi")
print("Sinä voitit", pelaajaPisteet, "kertaa")
print("Tietokone voitti", tietokonePisteet, "kertaa")
```

Kuva 26. Lopputehtävän esimerkkivastaus

8 JOHTOPÄÄTÖKSET

Valitsin opinnäytetyön aiheen koska, halusin tehdä ohjeet, joista voisi olla hyötyä muille ja halusin itse oppia lisää Python-ohjelmoinnista. Huomasin myös, kun mietin opinnäytetyön aihetta, että suomenkielisiä Python ohjeita löytyy erittäin vähän verkosta.

Itse opin opinnäytetyötä tehdessä uutena asiana ainakin moduulit ja kuinka paljon niitä on. Minun mielestäni moduulit olivat kumminkin tärkeä lisä työhön koska niitä on suuri kirjasto tarjolla ja niitä voi käyttää niin moneen eri asiaan. Mahdollisia lukijoita miettien yritin tehdä työn mahdollisimman selkeällä kielellä ja käyttää kuvia koodista avustamaan selityksiä. Itse en ainakaan opi niin helposti pelkästä teoriasta. Tämän takia yritinkin jättää teoriasta ylimääräisen pois ja toivon että, koodinpätkien ja tehtävien avulla saisi enemmän irti aiheesta. Tehtävät yritettiinkin suunnitella niin että eivät liian helppoja mutta kumminkin käyttäisivät kyseisen aiheen ja jo opittujen aiheiden perusteita. Mielestäni lopputehtävään sain sisällettyä lähes kaiken aikaisemmin opitun. Ehkä ainut miten lopputehtävää olisi voinut muokata olisi ollut input-komentoa käyttämällä pysäyttää ohjelma jokaisen silmukan lopussa. Mutta mielestäni input-komento ei oikein sopinut hyvin mihinkään aihealueeseen.

Mielestäni ehkä yksi hankalin asia oli valita mitä työhän haluaa sisällyttää, ja mitä ei. Olen vähän sitä mieltä, että ohjelmointikielissä on eroja mitkä asiat sisältyvät perusteisiin ja mitkä ovat vähän edistyneempiä. Tämä toki voi olla myös mielipide kysymys. Opinnäytetyössä verrattuna minun alkuperäiseen suunnitelmaani ei ole kuin muutama ero. Erot johtuvat juuri siitä, että mietin että kuuluuko tämä perusteisiin vai ei.

Aikataulua ajatellen pysyin aika lähellä alkuperäisessä aikataulussa, mutta opinnäytetyön aloitus pääsi alkuun vasta vähän myöhemmin syistä joita, en voinut itse hallita. Tämän takia en saanut aloitettua työtä ihan silloin kun aluksi oli tarkoitus. Mutta kun viimein sain aloitettua, kirjoittaminen eteni ihan hyvässä aikataulussa,

joten työ ei ollut loppujen lopuksi niin paljon myöhässä kuin olisi voinut olla, alkuperäisistä suunnitelmista ja vaikka en saanut aloitettua kirjoittamista ajallaan sain silti tehtyä työhön liittyviä asioita.

LÄHTEET

McKeown R. Why Is Python So Popular? An Introduction to The World's Favorite Programming Language. Viitattu 18.8.2021. https://learnpython.com/blog/why-is-python-so-popular/

Malik U. Advantages and Disadvantages of the Python Programming Language. Viitattu 20.8.2021. https://learnpython.com/blog/python-programming-advantages-disadvantages/

W3Schools 2021 A. Python print() Function. Viitattu 30.8.2021.

https://www.w3schools.com/python/ref_func_print.asp

Sturtz J. Variables in Python. Viitattu 2.9.2021. https://realpython.com/python-variables/

Agarwal D. Python if else. Viitattu 6.9.2021. https://www.geeksforgeeks.org/python-if-else/

W3Schools 2021 B. Python Conditions and If statements. Viitattu 7.9.2021. https://www.w3schools.com/python/python_conditions.asp

Python 2021. ForLoop. Viitattu 10.9.2021. https://wiki.python.org/moin/ForLoop

Learnpython 2021. Loops. Viitattu 10.9.2021. https://www.learnpython.org/en/Loops

Programiz 2021 A. Python while Loop. Viitattu 11.9.2021.https://www.programiz.com/python-programming/while-loop

Pythonbasics 2021. Functions in Python (With Examples) Viitattu 12.9.2021. https://pythonbasics.org/functions/

Programiz 2021 B. Python Function Arguments Viitattu 12.9.2021. https://www.programiz.com/python-programming/function-argument

Programiz 2021 C. Python Functions. Viitattu 12.9.2021. https://www.programiz.com/python-programming/function

Programiz 2021 D. Python Modules. Viitattu 13.9.2021. https://www.programiz.com/python-programming/modules

Tutorialsteacher 2021 A. Python – Built-in Modules Viitattu 14.9.2021. https://www.tutorialsteacher.com/python/python-builtin-modules

Tutorialsteacher 2021 B. Python – Random Module Viitattu 14.9.2021. https://www.tutorialsteacher.com/python/random-module