# Python-ohjelmointiharjoituksia

Näissä harjoituksissa opetellaan Pythonin tärkeimmät perusasiat ennen kuin siirrytään peliohjelmoinnin pariin. Harjoitukset etenevät helposta vaikeampaan.

Älä kopioi mallia sellaisenaan omaan tehtävääsi. Kirjoita ensin itse ja kokeile. Jos ohjelma ei toimi, lue virheilmoitus ja selvitä, missä kohtaa ongelma on.

## Harjoitus 0: Python IDLE:n käynnistäminen ja tiedoston tallentaminen

Python-harjoitukset tehdään Python IDLE -ohjelmassa. IDLE on Pythonin mukana tuleva yksinkertainen ohjelmointiympäristö.

**1. Avaa IDLE:** Avaa Windowsin Käynnistä-valikko ja kirjoita hakukenttään “IDLE”. Valitse IDLE (Python 3.x).

**2. Avaa editori:** Valitse IDLE:stä File → New File. Näkyviin tulee tyhjä koodieditori. Kirjoita varsinainen ohjelma tähän ikkunaan.

**3. Tallenna tiedosto heti alkuun:** Ennen kuin kirjoitat mitään, tallenna tiedosto valitsemalla File → Save As (tai Ctrl+S). Selaa tallennuspaikaksi OneDrive ja tee sinne kurssia varten oma kansio, esimerkiksi “Peliohjelmointi”, ja sen sisälle esimerkiksi “Python-harjoitukset”. Anna tiedostolle kuvaava nimi, esimerkiksi “harjoitus1.py”, ja paina Tallenna.

**4. Kirjoita koodi:** Kirjoita ohjelma editoriin. Voit käyttää samaa tiedostoa yhden harjoituksen tekemiseen, ja luoda seuraavaa harjoitusta varten uuden tiedoston samalla tavalla.

**5. Suorita ohjelma:** Kun olet kirjoittanut koodia, paina F5 tai valitse Run → Run Module. Ohjelman tuloste näkyy Shell-ikkunassa. Jos et ole vielä tallentanut, IDLE pyytää tallentamaan ennen suoritusta.

**6. Tarkista tallennuspaikka:** IDLE:n ikkunan otsikosta tai File → Save As -toiminnosta voit tarkistaa, että tiedosto on oikeasti OneDrivessa eikä esimerkiksi Downloads- tai Documents-kansiossa paikallisella koneella.

***Vinkki:*** *Tallenna ohjelma usein Ctrl+S-näppäinyhdistelmällä. OneDrive synkronoi tiedoston pilveen, jolloin työ ei jää vain koulun tietokoneelle.*

## Harjoitus 1: Tulostaminen

Aloitetaan kaikkein yksinkertaisimmasta asiasta: tekstin näyttämisestä ruudulla. Pythonin print-komento tulostaa eli näyttää tekstiä käyttäjälle.

Koodiin voi lisätä myös kommentteja #-merkillä. Kommentit ovat vain muistiinpanoja itselle, eikä Python suorita niitä. Voit käyttää niitä esimerkiksi silloin kun haluat muistaa myöhemmin, mitä jokin koodinpätkä tekee.

```
# Tämä on kommentti
print("Tervehdys!")
```

**Tehtävä:** Tulosta ruudulle teksti: “Tervetuloa Python-ohjelmoinnin pariin!”

## Harjoitus 2: Muuttujat ja tervehdys

Muuttuja on vähän kuin laatikko, johon voi tallentaa tiedon ja käyttää sitä myöhemmin uudelleen.

```
nimi = "Tarmo"
print("Hei ", nimi, "!")
```

**Tehtävä:** Luo muuttuja nimeltä pelaaja, anna sille jokin nimi ja tulosta “Tervetuloa peliin, *pelaaja*!”

## Harjoitus 3: Käyttäjän syöte

Ohjelma voi myös kysyä tietoa käyttäjältä. Tähän käytetään input()-komentoa.

```
nimi = input("Mikä on nimesi? ")
print("Hei", nimi, "!")
```

**Tehtävä:** Kysy käyttäjältä hänen lempipelinsä ja tulosta sen jälkeen “Sinun lempipelisi on *peli*.”

## Harjoitus 4: Luvut ja laskeminen

Komento input() palauttaa aina merkkijonon (str). Jos luvuilla halutaan laskea, syöte pitää ensin muuttaa numeroksi int()- tai float()-komennolla.

```
ika = int(input("Kuinka vanha olet? "))
print("Ensi vuonna olet", ika + 1, "vuotta vanha.")
```

**Tehtävä:** Kysy käyttäjältä pelaajan pisteet ja lisää niihin 100 pistettä. Tulosta uusi pistemäärä.

## Harjoitus 5: Yksinkertainen laskin

Python osaa tietysti myös laskea. Peruslaskutoimitukset ovat yhteenlasku (+), vähennyslasku (-), kertolasku (*), jakolasku (/) ja potenssi (**).

```
luku1 = float(input("Anna ensimmäinen luku: "))
luku2 = float(input("Anna toinen luku: "))

print("Summa:", luku1 + luku2)
print("Tulo:", luku1 * luku2)
```

**Tehtävä:** Tee ohjelma, joka kysyy pelaajan nykyiset pisteet ja yhden palkinnon antamat lisäpisteet. Tulosta lopullinen pistemäärä.

## Harjoitus 6: Ehtolauseet

Ohjelma voi toimia eri tavalla eri tilanteissa ehtolauseiden avulla. Tavallisimmat ehdot ovat:

| Ehto | Merkitys |
|---|---|
| `a == b` | a ja b ovat yhtä suuret |
| `a != b` | a ja b eivät ole yhtä suuret |
| `a < b` | a on pienempi kuin b |
| `a <= b` | a on pienempi tai yhtä suuri kuin b |
| `a > b` | a on suurempi kuin b |
| `a >= b` | a on suurempi tai yhtä suuri kuin b |

Ehtoja käytetään if-rakenteen avulla, esimerkiksi näin:

```
pisteet = int(input("Kuinka monta pistettä sait? "))

if pisteet >= 100:
    print("Hieno tulos!")
else:
    print("Harjoittele vielä!")
```

**Tehtävä:** Kysy pelaajalta pisteet. Jos pisteitä on vähintään 500, tulosta “Kultamitali!”. Jos pisteitä on vähintään 250, tulosta “Hopeamitali!”. Muuten tulosta “Pronssimitali!”.

***Vinkki:*** *Jos ensimmäinen ehto ei toteudu, saadaan seuraava ehto elif-komennolla.*

## Harjoitus 7: Numeroarvauspeli

Pythonissa on paljon valmiita ohjelmointikirjastoja, jotka tuovat lisää toimintoja käyttöön. Kokeillaan seuraavaksi random-kirjastoa, jolla voi arpoa satunnaisia lukuja.

```
import random

oikea_luku = random.randint(1, 10)
arvaus = int(input("Arvaa luku väliltä 1-10: "))

if arvaus < oikea_luku:
    print("Liian pieni!")
elif arvaus > oikea_luku:
    print("Liian suuri!")
else:
    print("Oikein!")
```

**Tehtävä:** Muokkaa peliä niin, että arvottava luku on väliltä 1–20.

## Harjoitus 8: Ensimmäinen funktio

Funktio on oma pieni koodinpätkä, jonka voi kutsua käyttöön aina uudelleen kirjoittamatta samaa koodia moneen kertaan.

```
def tervehdi(nimi):
    print("Tervetuloa,", nimi)

tervehdi("Tarmo")
```

**Tehtävä:** Tee funktio nimeltä aloita_peli(), joka tulostaa kolme riviä: pelin nimen, tervetuloviestin ja esimerkiksi “Onnea peliin!”. Kutsu funktiota kirjoittamalla sen nimi sulkeineen.

## Harjoitus 9: Funktio, joka palauttaa arvon

Funktio voi myös palauttaa tuloksen return-komennolla. Silloin funktion laskemaa arvoa voi käyttää suoraan myöhemmin koodissa.

```
def nelion_ala(sivu):
    ala = sivu * sivu
    return ala

tulos = nelion_ala(4)
print("Pinta-ala on", tulos)
```

**Tehtävä:** Tee funktio laske_pisteet(pisteet, bonus), joka palauttaa pisteiden ja bonuksen summan. Kutsu funktiota esimerkiksi arvoilla 300 ja 50 ja tulosta tulos.

## Harjoitus 10: for-silmukka

Silmukan avulla tietokone voi toistaa saman asian monta kertaa automaattisesti. Seuraava for-silmukka tulostaa luvut 1–5.

```
for i in range(1, 6):
    print(i)
```

**Tehtävä:** Tulosta kokonaisluvut 1–10 for-silmukalla. Sen jälkeen kokeile tulostaa jokaisen luvun perään teksti “pistettä”.

## Harjoitus 11: for-silmukka pelipisteillä

Silmukan sisällä voidaan myös kerätä tietoa, esimerkiksi laskea lukuja yhteen. Koodin rivi summa += luku lisää muuttujaan summa uuden luvun jokaisella kierroksella.

```
summa = 0
for luku in range(1, 6):
    summa += luku

print("Summa on", summa)
```

**Tehtävä:** Laske for-silmukalla, kuinka paljon pisteitä pelaaja saa, jos hän saa viidellä kierroksella aina 10 pistettä. Tulosta lopullinen pistemäärä.

## Harjoitus 12: while ja break

Joskus ohjelman pitää toistaa jotain niin kauan, kunnes jokin ehto täyttyy. Silloin käytetään while-silmukkaa. while True käynnistää silmukan, ja break-komennolla sen voi lopettaa halutussa kohdassa.

```
while True:
    vastaus = input("Kirjoita jotain (tai lopeta): ")

    if vastaus == "lopeta":
        break

    print("Kirjoitit:", vastaus)

print("Ohjelma päättyi.")
```

**Tehtävä:** Tee ohjelma, joka kysyy pelaajalta salasanaa niin kauan, että hän kirjoittaa sanan “python”. Kun salasana on oikein, tulosta “Oikein!” ja lopeta silmukka.

## Harjoitus 13: Turtle ja ensimmäinen piirros

Nyt siirrytään hetkeksi piirtämiseen. Pythonin turtle-kirjaston avulla voi ohjata ruudulla liikkuvaa “kynää”, joka piirtää jäljen liikkuessaan. forward() liikuttaa kynää eteenpäin ja left() kääntää sitä.

```
import turtle

wn = turtle.Screen()
t = turtle.Turtle()

for i in range(4):
    t.forward(100)
    t.left(90)

wn.mainloop()
```

**Tehtävä:** Muuta ohjelmaa niin, että turtle piirtää tasasivuisen kolmion. Mieti, kuinka monta kertaa silmukka toistetaan ja kuinka paljon turtle kääntyy.

***Vinkki:*** *Kolmiossa on kolme sivua. Yhden ulkokulman suuruus on 120 astetta.*

## Harjoitus 14: Turtle-funktio

Piirtämisen voi myös pakata funktioksi. Silloin samaa kuviota on helppo piirtää monta kertaa, vaikka eri kokoisena.

```
import turtle

wn = turtle.Screen()
t = turtle.Turtle()

def piirra_nelio(t, koko):
    for i in range(4):
        t.forward(koko)
        t.left(90)

piirra_nelio(t, 100)
wn.mainloop()
```

**Tehtävä:** Tee funktio piirra_kolmio(t, koko), joka piirtää kolmion. Kutsu funktiota vähintään kerran eri kokoisella arvolla.

***Vinkki:*** *Funktiolle voi antaa turtle-olion ja sivun koon parametreina samalla tavalla kuin mallissa.*

## Harjoitus 15: Turtle liikkuu näppäimillä

Nyt päästään jo lähelle oikeaa peliä: pelaaja voi ohjata hahmoa näppäimistöllä. Näppäimen painallukseen voidaan yhdistää oma funktio, ja tätä kutsutaan tapahtumaohjatuksi ohjelmoinniksi.

```
import turtle

wn = turtle.Screen()
wn.setup(width=600, height=400)

t = turtle.Turtle()
t.shape("turtle")

def liiku_ylos():
    y = t.ycor()
    t.sety(y + 20)

def liiku_alas():
    y = t.ycor()
    t.sety(y - 20)

wn.listen()
wn.onkeypress(liiku_ylos, "Up")
wn.onkeypress(liiku_alas, "Down")

wn.mainloop()
```

**Tehtävä:** Lisää ohjelmaan vasemmalle ja oikealle liikkuminen. Tee funktiot liiku_vasemmalle() ja liiku_oikealle() ja yhdistä ne vasempaan ja oikeaan nuolinäppäimeen.

***Vinkki:*** *xcor() kertoo turtlen x-koordinaatin. setx() muuttaa x-koordinaattia samalla tavalla kuin sety() muuttaa y-koordinaattia.*