Viikkotehtävät 6

Toistolauseiden ja ehtolauseiden yhdistäminen, break- ja continue -komennot

1. Pyydä ***silmukassa*** käyttäjältä 5 positiivista lukua. Näytä lopuksi ainoastaan **suurin käyttäjän syöttämä luku**.

   Vinkki: Tee silmukan ulkopuolelle muuttuja, joka pitää yllä tämän hetken isointa numeroa. Muuta numeroa aina silloin, **jos** uusi numero on isompi kuin aiempi numero.

   Lisätehtävä: Tarkista että käyttäjä syöttää datan oikeassa muodossa (positiivinen kokonaisluku)

   Esimerkki ohjelman toiminnasta:

```
Anna numero:
6
Anna numero:
4
Anna numero:
17
Anna numero:
15
Anna numero:
2
Suurin käyttäjän luku: 17
```

Vinkkejä: Tässä tehtävässä ei tarvitse kuin yhden toistolauseen ja yhden input()-lauseen!

Tehtävän tiedostonimi = ***exercise6_1.py***

Tyypillinen koodimäärä***: 6-10 riviä*** *(tyhjiä rivejä ja kommentteja ei lasketa)*

2. Tee ohjelma, jonka avulla käyttäjä voi tulostaa haluamansa luvun kertotaulun väliltä 1 – 10, niin monta kertaa kuin hän haluaa ohjelmaa käyttää. Kysy aluksi käyttäjältä, minkä luvun kertotaulun hän haluaa nähdä nyt, ja tulosta se sen jälkeen alla olevan kuvaesimerkin mukaisesti. Ohjelman toiminta loppuu **ainoastaan** silloin, jos käyttäjä syöttää **0:n**.

   Lisää ohjelmaasi myös ehtolause, joka tarkistaa onko käyttäjän syöttämä luku välillä 1 - 10. Jos näin ei ole, kysy lukua uudelleen niin kauan, kunnes käyttäjä syöttää sopivan arvon.

   Esimerkkejä ohjelman toiminnasta:

```
Minkä luvun kertotaulun haluat nähdä? (1-10). 0 lopettaa ohjelman.
7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70

Minkä luvun kertotaulun haluat nähdä? (1-10). 0 lopettaa ohjelman.
```

```
Minkä luvun kertotaulun haluat nähdä? (1-10). 0 lopettaa ohjelman.
16
Vääränlainen luku.

Minkä luvun kertotaulun haluat nähdä? (1-10). 0 lopettaa ohjelman.
```

   Vinkkejä: Tämä on helpointa tehdä sisäkkäisillä silmukoilla! Ensin while-silmukka, joka pyörittää ohjelmaa niin kauan kuin käyttäjä haluaa. while-silmukan sisällä, kun käyttäjältä on ensin input():lla kysytty mikä kertotaulu halutaan nähdä, tulostetaan kertotaulun rivit **for**-silmukalla. Enempää toistolauseita ei tarvita! Ohjelmassa tarvitsee olla myös vain yksi input()-lause! ☺

   Huom: **älä tulosta** kertotaulun rivejä 10:llä print()-kutsulla, vaan suorita print()-operaatio for-silmukassa!

   Tehtävän tiedostonimi = ***exercise6_2.py***
   Tyypillinen koodimäärä***: 10-16 riviä*** *(tyhjiä rivejä/kommentteja ei lasketa)*

3. Tee ohjelma, jonka avulla voidaan laskea tilastoa luokan oppilaiden koetuloksista. Ohjelman toiminta on seuraava:
    a) Kysytään oppilaiden lukumäärä. (Kannattaa testata aluksi pienellä oppilasmäärällä)

    b) Kysytään kunkin oppilaan koearvosana (silmukassa).

    c) Lasketaan keskiarvo arvosanoista ja tulostetaan se näytölle.

    d) Lisäksi tulostetaan näytölle keskimääräisestä arvosanasta sanallinen arvio silmukan jälkeen:

0 <= **keskiarvo** < 1	**Huono tulos**
1 <= **keskiarvo** < 2	**Heikko tulos**
2 <= **keskiarvo** < 3	**Tyydyttävä tulos**
3 <= **keskiarvo** < 4	**Hyvä tulos**
4 <= **keskiarvo** <= 5	**Kiitettävä tulos**

Lisätehtävä: Tallenna arvosanat listaan, ja hyödynnä sitä keskiarvon laskemisessa ja sanallisen arvion tulostamisessa.

Toinen lisätehtävä: Laske myös arvojen mediaani.

Kolmas lisätehtävä: Laske myös yleisin arvosana, eli moodi.
(**Vinkki**: hyödynnä listaa (list)!)

Vinkkejä: Älä pyöristä keskiarvoa silmukassa, vaan laske keskiarvo vasta silmukan jälkeen, jotta vältetään pyöristysvirheet.

Esimerkki ohjelman toiminnasta:

```
Opiskelijoiden lukumäärä:
5
Anna opiskelijan arvosana:
5
Anna opiskelijan arvosana:
3
Anna opiskelijan arvosana:
4
Anna opiskelijan arvosana:
1
Anna opiskelijan arvosana:
4
Keskiarvo: 3.4
Hyvä tulos
```

Tehtävän tiedostonimi = ***exercise6_3.py***

Tyypillinen koodimäärä***: 18-28 riviä*** *(tyhjiä rivejä ja kommentteja ei lasketa)*

4. Tee ohjelma, jossa on seuraavanlainen lista tunnisteita:

```
products = ["K1565_2017_ST7745", "T2432_2019_FE84",
            "T8551_2018_XA413", "T3345_2019_JK142",
            "Y51372_2019_HJ2", "Y76715_2017_AB3",
            "E98144_2018_21T", "T7733_2020_CE55",
            "E7614_2021_XZA784", "E9722_2017_MHE67",
            "Y82018_2020_FI95", "T61287_2021_IA293",
            "E9152_2019_TY5", "T774_2021_OB672"]
```

HUOM! Tästä listasta löytyy helposti kopioitava versio Moodlesta! Älä muokkaa listan dataa itse, vaan ohjelmasta täytyy löytyä yllä oleva lista sellaisenaan koodin alussa. Älä myöskään poista listasta mitään elementtiä koodilla.

Lista koostuu tilaustunnisteista, joista yksittäinen tunniste noudattaa seuraavaa kaavaa:

TILAUSKOODI_VUOSILUKU_LISÄTUNNISTE

Pyydä käyttäjältä vuosiluku, ja tulosta vain ne **TILAUSKOODIT** listasta, joiden vuosiluku täsmää käyttäjän antaman vuoden kanssa. Eli jos käyttäjä syöttää vuoden 2019, tulostetaan allekkain koodit ”T2432”, "T3345”, "Y51372" ja "E9152". Huomaa, että tilauskoodi on tapauksesta riippuen pituudeltaan 4-6 merkkiä pitkä. Lisätunniste on ylimääräinen tieto, jota ei tarvitse tämän tehtävän koodissa käsitellä.

Huom: **Tulosta ainoastaan tunnisteen TILAUSKOODI**, jätä muut asiat tulostamatta!

Vinkkejä: Hyödynnä split() –funktiota, jotta saat tiedot omiin muuttujiin ehtolausetta varten! (ks. materiaalit).

HUOM! osateksti (substring) ei sovellu tähän tehtävään hyvin, koska eri tietueet ovat tilanteesta riippuen eripituisia keskenään.

Huom: Pelkkä **if year in code** – tyyppinen ratkaisu **ei riitä**, sillä haettava vuosiluku saattaa olla käytössä myös jossain tilauskoodissa!
(esim. vuosi 2018 ja koodi Y82018_2020_FI95)

Tässä tehtävässä tarvitsee vain yhden for-silmukan!

Esimerkki ohjelman toiminnasta:

```
Minkä vuoden koodit haetaan?
2021
E7614
T61287
T774
```

Tehtävän tiedostonimi = ***exercise6_4.py***

Tyypillinen koodimäärä***: 12-18 riviä*** *(tyhjiä rivejä ja kommentteja ei lasketa)*

5. Käytä samaa tuotetunnistelistaa (**products)** kuin edellisessä tehtävässä.

   Tee ohjelma, joka kysyy käyttäjältä jonkin tuotekoodin (esim. E9722). Käy tämän jälkeen tuotetunnistelista läpi silmukassa. Jos etsitty koodi tulee silmukassa vastaan, tulosta tuotekoodin **pelkkä vuosiluku** sekä kutsu heti perään **break**-komentoa silmukan lopettamiseksi.

   Huom: break-komento on tässä tilanteessa hyödyllinen, sillä ohjelman tarkoitus on etsiä vain tietyn yksilöllisen tuotekoodin vuosiluku. Kun vuosiluku löytyy, **on turhaa enää etsiä pidemmälle, koska haluttu tieto on jo löydetty**! **break**-komento lopettaa silmukan suorituksen ennenaikaisesti suorituskyvyn parantamiseksi.

   Vinkkejä: Hyödynnä split() –funktiota, jotta voit käsitellä tuotekoodin eri osia omissa muuttujissaan! (ks. materiaalit).

   Huom: Tämän tehtävän koodi on hyvin lähellä edellistä tehtävää! **Ohjelman tulee tulostaa vain 1 vuosiluku!**

   HUOM! osateksti (substring) ei sovellu tähän tehtävään hyvin, koska eri tietueet ovat tilanteesta riippuen erityisiä keskenään.

   Huom: Pelkkä **if user_code in order** – tyyppinen ratkaisu **ei riitä**, sillä haettava tuotekoodi saattaa olla käytössä myös muualla tilauskoodissa!
   (esim. koodi T774 -> K1565_2020_ST7745)

   Tässä tehtävässä tarvitsee vain yhden for-silmukan!

   Esimerkki ohjelman toiminnasta:

```
Syötä tuotekoodi:
Y51372
Tilauksen vuosiluku: 2019
```

Tehtävän tiedostonimi = ***exercise6_5.py***

Tyypillinen koodimäärä***: 12-18 riviä*** *(tyhjiä rivejä ja kommentteja ei lasketa)*

Lisätehtäviä:

Huom: Hyvä arvosana ei tarvitse kaikkien lisätehtävien tekemistä. Voit valita itsellesi mieluisat tehtävät!

6. Tee uusi ohjelma, ja luo siihen lista nimeltä ”**basket**”. Lisää siihen seuraavat sisältö:
    - "omena"
    - "banaani"
    - "kirsikka"
    - "porkkana"
    - "peruna"
    - "tomaatti"
    - "kaali"

    Kysy tämän jälkeen käyttäjältä jokin sana. Tulosta tämän jälkeen **basket**-listan sisältö allekkain siten, että **jätät tulostamatta käyttäjän antaman sanan** (esim. **continue**-komennon avulla).

    Jos valittua sanaa ei löydy listasta, ilmoita käyttäjälle: "***Sanaa ei löytynyt.", äläkä tällöin tulosta basket-muuttujan sisältöä ollenkaan.***

    Vinkkejä: Tämän voi tehdä usealla eri tavalla. Suoraviivaisin ratkaisu on käydä lista läpi silmukalla ja käyttää **continue** -komentoa silloin, jos valittu sana tulee kohdalle (esim. **if word == userword**) => ***continue)***. Jos sana ei tullut kohdalle, tulostetaan vuorossa oleva sana sellaisenaan

    *Muista käyttää yllä olevaa if-lausetta ennen sanan tulostamista, jotta **continue** –komento kerkeää käsitellä asian ennen tulostamista.*

    Tehtävän voi myös ratkaista myös **if/else** –rakennetta tai kokoelmafunktioita käyttämällä.

Lisätehtävä: Jos käyttäjä syöttää sanan sijasta numeron, tulosta muutoin kaikki listan sisältö, mutta jätä tulostamatta annetulla järjestysnumerolla oleva sana listasta. Eli jos käyttäjä syöttää vaikkapa numeron 3, jätetään tässä tapauksessa ***kirsikka*** tulostamatta.

> **Vinkki:** käytä **text.isnumeric()** –funktiota hyödyksi, jotta tiedät syöttikö käyttäjä tekstin vai numeron! Muista: kokoelman ensimmäinen indeksi on 0.

Esimerkkejä ohjelman toiminnasta:

```
Syötä sana:
porkkana
omena
banaani
kirsikka
peruna
tomaatti
kaali
```

```
Syötä sana:
banaani
omena
kirsikka
porkkana
peruna
tomaatti
kaali
```

```
Syötä sana:
papukaija
Sanaa ei löytynyt listasta.
```

Tehtävän tiedostonimi = ***exercise6_6.py***
Tyypillinen koodimäärä***: 8-20 riviä*** *(tyhjiä rivejä ja kommentteja ei lasketa)*

7. Tee ohjelma, joka etsii käyttäjän määrittelemältä lukualueelta lukua, joka on jaollinen viidellä ja seitsemällä. Ohjelma toimii siten, että käyttäjältä kysytään hakualueen ala- ja yläraja. Tämän jälkeen tulostetaan ohjelmassa jokaisesta luvusta tieto, onko se jaollinen 5:lla ja 7:lla vai ei.

   Ohjelma tulee toteuttaa siten, että mikäli luku ei ole jaollinen viidellä, siirrytään suoraan testaamaan seuraavaa lukua. Etsintä lopetetaan heti, kun ohjelma löytää sopivan luvun.

   Lopullisen vastauksen tulisi konsolissa näyttää esim. tältä:

   Anna alueen alaraja: 15
   Anna alueen yläraja: 100
   15 ei ole jaollinen seitsemällä, hylätään.
   16 ei ole jaollinen viidellä, hylätään.
   17 ei ole jaollinen viidellä, hylätään.
   18 ei ole jaollinen viidellä, hylätään.
   19 ei ole jaollinen viidellä, hylätään.
   20 ei ole jaollinen seitsemällä, hylätään.
   21 ei ole jaollinen viidellä, hylätään.
   22 ei ole jaollinen viidellä, hylätään.
   23 ei ole jaollinen viidellä, hylätään.
   24 ei ole jaollinen viidellä, hylätään.
   25 ei ole jaollinen seitsemällä, hylätään.
   26 ei ole jaollinen viidellä, hylätään.
   27 ei ole jaollinen viidellä, hylätään.
   28 ei ole jaollinen viidellä, hylätään.
   29 ei ole jaollinen viidellä, hylätään.
   30 ei ole jaollinen seitsemällä, hylätään.
   31 ei ole jaollinen viidellä, hylätään.
   32 ei ole jaollinen viidellä, hylätään.
   33 ei ole jaollinen viidellä, hylätään.
   34 ei ole jaollinen viidellä, hylätään.
   Luku 35 on jaollinen 5:llä ja 7:llä.
   Lopetetaan haku.
   >>> ESIM 2:
   Anna alueen alaraja: 11
   Anna alueen yläraja: 13
   11 ei ole jaollinen viidellä, hylätään.
   12 ei ole jaollinen viidellä, hylätään.
   13 ei ole jaollinen viidellä, hylätään.
   Alueelta ei löytynyt sopivaa arvoa.

   Tehtävän tiedostonimi = ***exercise6_7.py***

   Tyypillinen koodimäärä***: 20-36 riviä*** *(tyhjiä rivejä ja kommentteja ei lasketa)*

8. Tee ohjelma, **joka tuottaa halutun mittaisia turvallisia salasanoja**. Pyydä ensin käyttäjältä merkkimäärä, kuinka pitkän salasanan hän haluaa. Älä huoli alle 8:n merkin pituisia salasanoja. Rakenna tämän jälkeen (käyttämällä silmukkaa) merkkimäärää vastaavan pituinen salasanaehdotus, joka täyttää seuraavat ehdot:

    a) Salasanassa on vähintään yksi pieni kirjain
    b) Salasanassa on vähintään yksi iso kirjain
    c) Salasanassa on vähintään yksi numero
    d) Salasanassa on vähintään jokin seuraavista erikoismerkeistä:
    - _ ~ ? * $ # ! + % @

    Huom: Tämän tehtävän voi ratkaista monella eri tavalla!

Tehtävän tiedostonimi = ***exercise6_8.py***

Tyypillinen koodimäärä***: 16-32 riviä*** *(tyhjiä rivejä ja kommentteja ei lasketa), tähän voi mennä enemmänkin rivejä, miäli toteutus on erityisen monipuolinen!*

9. **Haastava lisätehtävä!** Tee ohjelma, joka laskee autokaupalle käytetyn auton myyntihinnan seuraavien tietojen perusteella (kysy nämä tiedot käyttäjältä):
    - Auton alkuperäinen hinta
    - Auton valmistusvuosi
    - Ajokilometrit
    - Auton valmistajan hintakategoria (1 vai 2)
    - Onko kyseessä tuontiauto (k/e)

    Ohjelma laskee käytetylle autolle myyntihinnan seuraavalla logiikalla:

    - **Jos kyseessä on hintakategoria 2:**
        - Jos ajokilometrit ovat keskimäärin 30000 km / vuosi tai enemmän:
            - Auton arvo laskee 10% jokaista vuotta kohden ensimmäisen viiden ikävuoden aikana, jonka jälkeen hinta laskee 7% jokaista vuotta kohden
        - Jos ajokilometrit ovat alle 30000km / vuosi:
            - Auton arvo laskee 8% jokaista vuotta kohden ensimmäisen viiden ikävuoden aikana, jonka jälkeen hinta laskee 5% jokaista vuotta kohden
        - Auton hinta ei koskaan laske kuitenkaan 12%:n alle alkuperäisestä hinnasta

    - **Jos kyseessä on hintakategoria 1:**
        - Jos ajokilometrit ovat keskimäärin 30000 km / vuosi tai enemmän:
            - Auton arvo laskee 7% jokaista vuotta kohden ensimmäisen viiden ikävuoden aikana, jonka jälkeen hinta laskee 4% jokaista vuotta kohden
        - Jos ajokilometrit ovat alle 30000km / vuosi:
            - Auton arvo laskee 5% jokaista vuotta ensimmäisen viiden ikävuoden aikana, jonka jälkeen hinta laskee 3% jokaista vuotta kohden
        - Auton hinta ei koskaan laske kuitenkaan 18%:n alle alkuperäisestä hinnasta

    - Tuontiautojen lopulliseen hintaan lisätään 24% veroa

Kun ohjelma on laskenut myyntihinnan autolle, kysytään käyttäjältä, haluaako hän syöttää uuden auton tiedot ohjelmaan. Mikäli hän vastaa ’e’, lopetetaan ohjelman suoritus ja kiitetään ohjelman käytöstä. Muussa tapauksessa ohjelma aloitetaan alusta.

Esimerkkejä ohjelman toiminnasta (oletuksena että kuluva vuosi on 2023):

Alkuperäinen hinta: 50000€ **Valmistusvuosi:** 2016 **Mittarilukema:** 140000km **Hintakategoria:** 2 **Tuontiauto:** Kyllä **Lopullinen hinta:** 36878 €	**Alkuperäinen hinta:** 35000€ **Valmistusvuosi:** 2019 **Mittarilukema:** 140000km **Hintakategoria:** 1 **Tuontiauto:** Ei **Lopullinen hinta:** 26181 €
Alkuperäinen hinta: 50000€ **Valmistusvuosi:** 2016 **Mittarilukema:** 250000km **Hintakategoria:** 2 **Tuontiauto:** Ei **Lopullinen hinta:** 25535 €	**Alkuperäinen hinta:** 40000€ **Valmistusvuosi:** 2014 **Mittarilukema:** 350000km **Hintakategoria:** 1 **Tuontiauto:** Kyllä **Lopullinen hinta:** 29307 €

Tehtävässä ei niinkään arvioida laskutoimituksen 100% hintatarkkuutta, vaan ohjelman rakennetta ja logiikkaa!
Riittää, kun laskentatapa osuu suurin piirtein oikeaan kokoluokkaan.

Tehtävän tiedostonimi = ***exercise6_9.py***

Tyypillinen koodimäärä***: 40-60 riviä*** *(tyhjiä rivejä/kommentteja ei lasketa)*