Viikkotehtävät 5

Toistolauseet ja kokoelmat (**for, while - list, tuple, dict**)

1. a) Toteuta ohjelma, joka tulostaa luvut 1-50 allekkain.
   Käytä **while**-toistolausetta.

   b) Tee sama kuin edellisessä, mutta käytä **for** -toistolausetta.

   c) Tee silmukka, joka laskee kaikkien numeroiden summan
   väliltä 1 – 30. **Tulosta ainoastaan summan lopputulos, älä jokaista välivaihetta**. Käytä joko for- tai while- toistolausetta perustuen siihen, kumpaa pidät tähän tilanteeseen sopivampana.

   d) Tee silmukka, joka tulostaa kaikki vuosiluvut väliltä 2005-2010 välilyönnillä eroteltuna ***samalle*** riville. Käytä joko for- tai while- toistolausetta perustuen siihen, kumpaa pidät tähän tilanteeseen sopivampana.

   Esimerkki ohjelman toiminnasta:

```
40
41
42
43
44
45
46
47
48
49
50
Summa: 465
2005 2006 2007 2008 2009 2010
```

Vinkki: Tässä tehtävässä ei välttämättä tarvitse yhtään ehtolausetta!

Tehtävän tiedostonimi = ***exercise5_1.py***

Tyypillinen koodimäärä***: 14-20 riviä*** *(tyhjiä rivejä ja kommentteja ei lasketa)*

2. Kysy käyttäjältä 12 kertaa kuukauden sademäärä **silmukassa** (eli käytännössä kysytään vuoden jokaisen kuukauden sademäärä). Laske ja tulosta tämän jälkeen sademäärien keskiarvo näytölle ***pyöristettynä yhteen desimaaliin***.

   Huom: Käytä silmukkaa sademäärien kysymisessä! (esim. for-silmukka sopii hyvin). Ks. toistolauseiden materiaalien sivu 46.

   Vinkkejä: keskiarvo = kokonaissademäärä / kuukausien määrä

   Kokonaissademäärää kannattaa kasvattaa for-silmukassa!
   (vrt. tehtävä 5-1, kohta c))

   Tässä tehtävässä tarvitsee vain yhden toistolauseen!

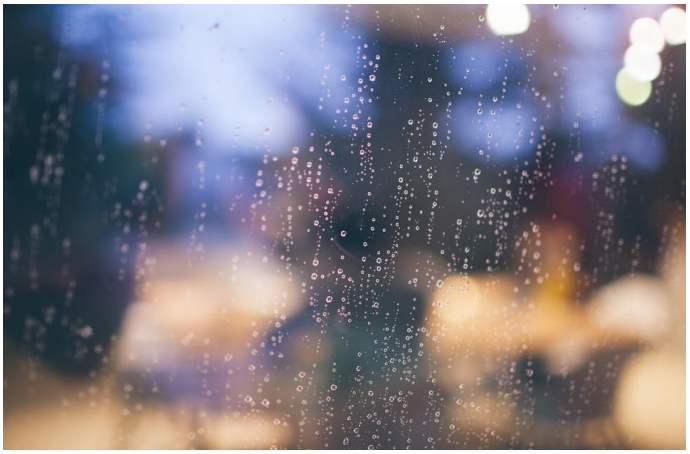
   Huom! Laske keskiarvo vasta silmukan ***jälkeen***, koska muutoin voi käydä niin, että tulokseen tulee pyöristysvirhe, jos keskiarvo lasketaan silmukan sisällä 12 kertaa.

   Esimerkki ohjelman toiminnasta:

```
60
Anna kuukauden sademäärä:
53
Anna kuukauden sademäärä:
47
Vuoden keskiarvo on n. 50.8 mm
```

Tehtävän tiedostonimi = ***exercise5_2.py***

Tyypillinen koodimäärä***: 7-12 riviä*** *(tyhjiä rivejä ja kommentteja ei lasketa)*



3. Tee ohjelma, joka kysyy käyttäjältä **ympyrän säteen** ja laskee sen jälkeen ympyrän pinta-alan sekä tulostaa sen näytölle. **Kysy lopuksi haluaako käyttäjä jatkaa ohjelman käyttöä (k/e)**.

   Jos käyttäjä syöttää **jotain muuta** kuin '***e***', aloitetaan ohjelma alusta (kysytään sädettä jne.).

   Jos käyttäjä syöttää '***e***', ohjelman suoritus lopetetaan.

   Pyöristä pinta-alan tulos kahteen desimaaliin.

   Muistutus: Ympyrän pinta-ala: πr^2

   Esimerkki ohjelman toiminnasta:

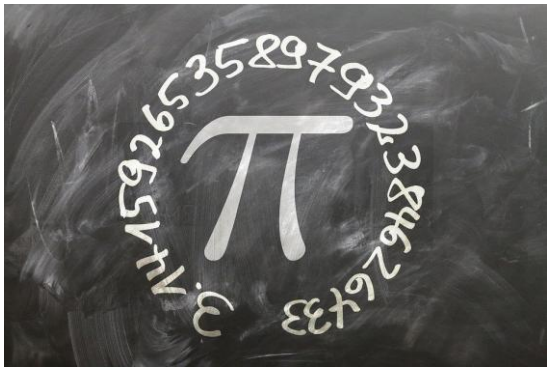
```
Anna säde:
15
Ympyrän pinta-ala: 706.86
Haluatko jatkaa? (k/e)
k
Anna säde:
7
Ympyrän pinta-ala: 153.94
Haluatko jatkaa? (k/e)
e
```

   Vinkki: Ks. while/running –esimerkki materiaaleista!

   Tässä tehtävässä tarvitsee vain yhden toistolauseen!

   Tehtävän tiedostonimi = ***exercise5_3.py***

   Tyypillinen koodimäärä***: 10-16 riviä*** *(tyhjiä rivejä ja kommentteja ei lasketa)*



4. Tee ohjelma, jossa on seuraavanlainen yksinkertainen lista (**list**):
    - Rooma
    - Ateena
    - Tukholma
    - Lontoo
    - Dublin
    - Pariisi

Tulosta lista **käyttämällä silmukkaa**. Tulosta myös jokaisen rivin rivinumero alkaen 1:stä.

Vinkki: voit hyödyntää silmukan indeksinumeroa rivinumeron tulostamisessa (ks. amount –esimerkki materiaaleista)

Pieni lisätehtävä: Laita lista Pythonin avulla aakkosjärjestykseen ennen tulostamista.

Esimerkki ohjelman toiminnasta:

```
1: Rooma
2: Ateena
3: Tukholma
4: Lontoo
5: Dublin
6: Pariisi
```

Vinkkejä: Tässä tehtävässä tarvitsee vain yhden toistolauseen!

Huom: Tehtävässä tulee käyttää pelkästään yksinkertaista listaa (esim. ***cities = ['Rooma', 'Ateena' …]*** jne.), **älä rakenna dataa sellaiseen muotoon, jossa esim. numerointi tulisi helpommin mukana** (tämä siksi, että tositilanteessa emme yleensä voi vaikuttaa listan rakenteeseen, vaan se tulee siinä muodossa kuin se tulee käytettäväksi koodiin)

Tehtävän tiedostonimi = ***exercise5_4.py***
Tyypillinen koodimäärä***: 4-12 riviä*** *(tyhjiä rivejä/kommentteja ei lasketa)*

5. Tee ohjelma, jossa on **tuple**, joka sisältää suomeksi eri kuukausien nimet (Tammikuu, Helmikuu jne.). Pyydä käyttäjältä kuukauden numero (1-12) ja tulosta numeroa vastaava kuukauden nimi **tuplen** avulla.

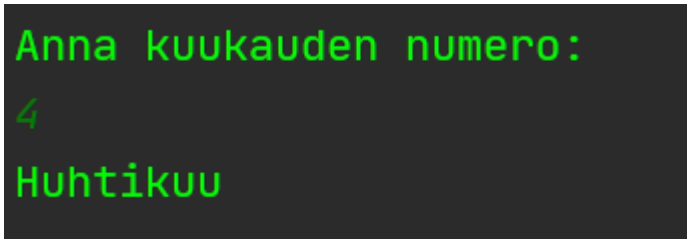
   Huom: kokoelmat alkavat 0:sta, ei 1:stä!

   Vinkkejä: Kun käyttäjän antama numero on muutettu numeroksi int() -avulla, siitä voidaan vähentää 1 jotta kokoelman indeksi täsmää kuukauden nimen kanssa!

   Teknisesti on myös mahdollista lisätä "tyhjä" arvo ensimmäiseksi tuplen elementiksi, **mutta se ei ole optimaalinen tapa**, ja voi joidenkin koodareiden mielestä olla epäselvä ratkaisu.

   Vinkki: Tässä tehtävässä ei tarvitse yhtään toistolausetta!

   Esimerkki ohjelman toiminnasta:



   Tehtävän tiedostonimi = ***exercise5_5.py***

   Tyypillinen koodimäärä***: 3-6 riviä*** *(tyhjiä rivejä/kommentteja ei lasketa)*

6. Tee ohjelma, jossa on seuraavanlainen ***dictionary*** (**dict**):
   - **name**: ”Pulp Fiction”
   - **year**: 1994
   - **director**: ”Quentin Tarantino”
   - **genre:** ”Rikos, Draama”
   - **duration:** 154

   Tulosta tämän jälkeen dictionaryn sisältö allekkain **ilman silmukkaa**.

   ks. Toistolauseet -materiaaleista sivu 73!

Esimerkki ohjelman toiminnasta:

```
Pulp Fiction
1994
Quentin Tarantino
Rikos, Draama
154
```

Vinkki: Tässä tehtävässä ei tarvitse yhtään toistolausetta!

Tehtävän tiedostonimi = ***exercise5_6.py***

Tyypillinen koodimäärä***: 8-16 riviä*** *(tyhjiä rivejä ja kommentteja ei lasketa)*

Lisätehtäviä!

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

7. Tee ohjelma, jossa on lista, joka sisältää seuraavat tuotetunnisteet teksteinä:
    - **”PHILIPS_Vedenkeitin_HD4646_2020_09_21_C_1”**
    - **”KENWOOD_Yleiskone_KVL8300S_2015_09_22_C_3”**
    - **”BOSCH_Tehosekoitin_MMB65G5M_2016_05_07_C_3”**
    - **”WHIRLPOOL_Mikroaaltouuni_MCP345WH_2019_01_15_C_1”**
    - **”ROSENLEW_Pakastin_RPP2330_2017_01_29_C_2”**
    - **”ELECTROLUX_Jääkaappi_ERF4114AOW_2017_11_07_C_2”**

    Huom: älä pilko tuotetunnisteita valmiiksi itse, vaan säilytä tuotetunnisteet sellaisenaan listassa!

    Yksittäinen tuotetunniste koostuu seuraavasta kaavasta:

    VALMISTAJA_TUOTENIMI_MALLI_VUOSI_KUUKAUSI_PAIVA_C_KATEGORIA

    Kategorioiden numerot ovat välillä 1-3, ja ne vastaavat seuraavia selitteitä:

    1 = Pienlaitteet
    2 = Kylmälaitteet
    3 = Sekoittimet

    Käsittele jokainen sarjatunniste silmukassa, ja irrota siitä kaikki tarvittavat tiedot omiin muuttujiinsa. (ks. materiaalit: Toistolauseet) Tulosta jokainen tuote seuraavassa muodossa:

    Valmistaja: PHILIPS
    Nimi ja malli: Vedenkeitin (HD4646)
    Kategoria: Pienlaitteet
    Lisäyspäivämäärä: 21.9.2020

    Tulosta myös tyhjä rivi jokaisen tuotekuvauksen jälkeen.

Vinkkejä: Kategorioiden selitteet kannattaa säilyttää omassa listassaan, esim.
categories = ["Muut", "Pienlaitteet", "Kylmälaitteet", "Sekoittimet"]

Tämän jälkeen kategorian numeron avulla on mahdollista hakea suoraan listasta tarvittava kategoria.
esim.

```
# jos kategoria on esim. 1
c = 1
description = categories[c]
```

Tämä rakenne toimii joustavasti vaikka kategorioita olisi kymmeniäkin!

Huom: älä pilko tuotetunnisteita valmiiksi itse, vaan säilytä tuotetunnisteet sellaisenaan listassa!

Muista käyttää split() –funktiota, jotta saat tuotetunnisteiden tiedot omiin muuttujiin silmukassa!

Tehtävän tiedostonimi = ***exercise5_7.py***
Tyypillinen koodimäärä***: 15-40 riviä*** *(tyhjiä rivejä ja kommentteja ei lasketa)*

8. Tee ohjelma, joka pyytää käyttäjältä sanan. Pyydä tämän jälkeen käyttäjältä salauksessa tarvittava siirtoluku. Suorita annetulle sanalle ns. Caesarin salakirjoitus käyttämällä siirtolukua, ja tulosta lopputulos.

   Lisätietoa Caesarin salakirjoituksesta:
   https://fi.wikipedia.org/wiki/Caesarin_salakirjoitus

   Esim. jos alkuperäinen teksti on **”vesilasi”** ja siirtoluku on 3, silloin salakirjoitettu versio on **”yhvlodvl”**.

   Vinkki: Tähän on useita tapoja miten tämän kanssa voi edetä, mutta asian kanssa voi edetä esim. seuraavien funktioiden avulla:

   esim.

```
# tämä antaa annetun kirjaimen sijainnin merkistössä (luku)
index = ord("a")

# tämä antaa merkin annetulla merkistön sijoitusluvulla
char = chr(99)
```

P.S. Caesarin salakirjoitus on erittäin helppo murtaa todellisessa elämässä, joten se ei sovellu tietoturvatarkoituksiin, mutta se on hyvä ohjelmointiharjoitus!

Tehtävän tiedostonimi = ***exercise5_8.py***

Tyypillinen koodimäärä***: 10-20 riviä*** *(tyhjiä rivejä ja kommentteja ei lasketa)*

9. Pythonissa on olemassa ominaisuus nimeltä ***list comprehension***, jonka tarkoituksena on helpottaa monimutkaisia hakuja, suodatuksia ja toimenpiteitä jotka kohdistuvat erilaisiin listoihin tai kokoelmiin.

   Ks. esimerkkejä:
   - https://www.programiz.com/python-programming/list-comprehension
   - https://www.digitalocean.com/community/tutorials/understanding-list-comprehensions-in-python-3

   Tässä lisätehtävässä:

   1. Tee ohjelmassa lista, jossa on vähintään 10 eri suomalaista etunimeä

   2. Käytä ***list comprehensionia***, ja tulosta ainoastaan ne nimet **jotka täyttävät KAIKKI seuraavat ehdot yhtä aikaa (Huom: pelkät erilliset list comprehension-esimerkit eivät riitä, vaan lopputuloksena tulee olla lista, josta kaikki on suodatettu!)**:
      - Nimet joissa ei ole s-kirjainta
      - Nimet joissa ei ole e-kirjainta
      - Nimet jotka ovat alle 8 merkkiä pitkiä
      - **Lisätehtävä:** Nimet joissa on maksimissaan 2 vokaalia (haastava!)
        - Voit käyttää tässä Python 3:ssa olevia funktioita:
          1. sum(), list(), map()

Vinkki:
Yksi tapa tehdä tämä on suodattaa dataa sääntö kerrallaan. Eli jos ensimmäinen list comprehension –lause suodattaa dataa muuttujaan **result**, voit seuraavassa list comprehension –lauseessa käsitellä **result**-muuttujaa **alkuperäisen datan sijasta**! Tällä tavalla voit käsitellä datan osissa, ja lopputuloksena on vain yksi lista henkilöistä, joiden nimissä ei ole s- tai e-kirjainta sekä ovat pituudeltaan alle 8 merkkiä pitkiä.

Tehtävän tiedostonimi = ***exercise5_9.py***

Tyypillinen koodimäärä***: 12-24 riviä*** *(tyhjiä rivejä ja kommentteja ei lasketa)*