Harjoitus 4 (viikko 39)

- Mikäli tehtävissä on jotain epäselvää, laita sähköpostia vastuuopettajalle (jorma.laurikkala@tuni.fi).
- Tee ohjelman alkuun kommentti, jossa on harjoituksen ja tehtävän numero sekä nimesi ja sähköpostisoitteesi. Sisennä neljällä välilyönnillä. Muista myös kaikki muut hyvät tavat (katso luentorungon 11. luku).
- Ensi viikolla pidettävissä mikroharjoituksissa ja Luupin koodauspajassa saa apua ongelmakohtiin. Keskiviikon mikroharjoitusryhmässä avustetaan hieman enemmän kuin muissa ryhmissä. Mikroharjoituksissa autetaan myös Pythonin asennuksessa omalle kannettavalle tietokoneelle.
- Palauta vastauksesi WETO-järjestelmään (https://wetodev.sis.uta.fi/) viimeistään ensi viikon torstaina 26.9. klo 12.00 (keskipäivä).
- WETO tarkistaa kaikki tämän harjoituskerran tehtävät automaattisesti. Varmista, että ohjelmasi toimii täsmälleen esimerkkien mukaisesti, koska tarkistus tehdään opiskelijan ja malliohjelman tulosteita vertailemalla. Huomaa, että rivien alkuun tai loppuun ei tulosteta välilyöntejä ja että kaikki tulostettavat rivit – viimeinen rivi mukaan lukien – päätetään rivinvaihtoon. Lisätietoja tarkistuksesta on saatavilla kurssisivujen *Harjoitukset* | *Ratkaisujen tarkistus* -kohdassa. Ota yhteyttä harjoitusryhmäsi vetäjään, jos et keksi järjellisessä ajassa miksi WETO hylkää palautuksesi.

1. Lausekielinen ohjelmointi I -kurssin arvosana määräytyy tenttipisteiden (0–24 kpl) ja tenttipisteisiin lisättävien hyvityspisteiden (0–3 kpl) summan perusteella seuraavasti: 12–14 pistettä → 1 (välttävä), 15–17 pistettä → 2 (tyydyttävä), 18–20 pistettä → 3 (hyvä), 21 tai 22 pistettä → 4 (kiitettävä) ja vähintään 23 pistettä → 5 (erinomainen). Hyvityspisteet huomioidaan vasta, kun opiskelija on suorittanut tentin hyväksyttävästi eli saanut tentistä vähintään 12 pistettä.

   Kirjoita Python-ohjelma, joka lukee käyttäjältä tenttipisteiden ja hyvityspisteiden lukumäärät sekä laskee ja tulostaa pisteiden summaa vastaavan arvosanan. Ohjelma tulostaa virheilmoituksen "I cannot grade you.", jos tenttipisteet eivät ole välillä 12–24 tai hyvityspisteet eivät ole välillä 0–3. Hyvityspisteet kysytään, vaikka tenttipisteet olisivat epäkelvot. Käytä ohjelmassa loogisia operaattoreita. Palauta ohjelma *grader.py*-nimisessä tiedostossa.

   Esimerkki ohjelman toiminnasta, kun syötteet ovat 22 ja 1:

   ```
   Hello! I am a grader.
   Please, enter exam points:
   22
   Please, enter bonus points:
   1
   Your grade is 5.
   ```

   Esimerkki ohjelman toiminnasta, kun syötteet ovat 15 ja 3:

   ```
   Hello! I am a grader.
   Please, enter exam points:
   15
   Please, enter bonus points:
   3
   Your grade is 3.
   ```

   Esimerkki ohjelman toiminnasta, kun syötteet ovat 11 ja 1:

   ```
   Hello! I am a grader.
   Please, enter exam points:
   11
   Please, enter bonus points:
   1
   I cannot grade you.
   ```

2. Tee Python-ohjelma, joka lukee käyttäjältä kaksi merkkiä ja päättelee onko kyseessä kaari-, aalto-, haka- tai kulmasulkeiden pari vai ei. Päättelyn tulos kerrotaan käyttäjälle. Sulkeet katsotaan pariksi, jos parin ensimmäinen merkki on “avaava” sulje ja jälkimmäinen merkki on “sulkeva” sulje. Näin syötteet "(" ja ")", "{" ja "}", "[" ja "]" ja "<" ja ">" ovat pareja. Toisaalta esimerkiksi syötteet ")" ja "(" eivät muodosta paria, koska sulkeiden järjestys on väärä.

   Käytä ohjelmassa loogisia operaattoreita. Palauta ohjelma *pairs.py*-nimisessä tiedostossa.

   Esimerkki ohjelman toiminnasta, kun käyttäjän syötteet ovat "{" ja "}":

   ```
   Hello! I find pairs.
   Enter the first character:
   {
   Enter the second character:
   }
   Characters "{" and "}" are a pair.
   ```

   Esimerkki ohjelman toiminnasta, kun käyttäjän syötteet ovat "X" ja "X":

   ```
   Hello! I find pairs.
   Enter the first character:
   X
   Enter the second character:
   X
   Characters "X" and "X" are not a pair.
   ```

   Esimerkki ohjelman toiminnasta, kun käyttäjän syötteet ovat "(" ja "s":

   ```
   Hello! I find pairs.
   Enter the first character:
   (
   Enter the second character:
   s
   Characters "(" and "s" are not a pair.
   ```

3. Tee Pythonilla ohjelma, joka lukee käyttäjältä kaksi merkkijonoa ja kertoo alla annettujen esimerkkien mukaisesti ovatko ne samat vai erilaiset, kunnes käyttäjä antaa molemmiksi syötteiksi merkkijonon "xyz".

   Määrittele ohjelman pysäyttävä merkkijono vakiona ohjelman alussa. Nimeä vakiosi siten, että siitä ei käy ilmi suoraan sen arvo, vaan vakion tarkoitus ohjelmassa. XYZ on siten huono nimi, vaikka se on annettu tyylillisesti oikein pelkästään suuria kirjaimia käyttäen. Muista käyttää vakiota ohjelmassa, sillä pelkästä vakion määrittelystä ei ole mitään hyötyä. Opettaja antaa ratkaisulle pisteen vain, jos vakio on määritelty ja sitä on lisäksi käytetty ohjelmassa.

   Palauta ohjelma *string_comparator.py*-nimisessä tiedostossa.

   Esimerkki ohjelman toiminnasta, kun syöteparit ovat "abba" ja "abba", "Aargh!" ja "aargh!", "cat" ja "dog" sekä "xyz" ja "xyz":

```
Hello! I compare strings.
Please, enter the first string:
abba
Please, enter the second string:
abba
"abba" is equal to "abba".
Please, enter the first string:
Aargh!
Please, enter the second string:
aargh!
"Aargh!" is different from "aargh!".
Please, enter the first string:
cat
Please, enter the second string:
dog
"cat" is different from "dog".
Please, enter the first string:
xyz
Please, enter the second string:
xyz
```

Esimerkki ohjelman toiminnasta, kun syöteparit ovat "xyz" ja "xyz":

```
Hello! I compare strings.
Please, enter the first string:
xyz
Please, enter the second string:
xyz
```

4. Kirjoita Pythonilla ohjelma merkkijonojen pituuksien vertailuun. Ohjelma lukee käyttäjältä kaksi merkkijonoa ja kertoo alla olevien esimerkkien mukaisesti onko ensimmäinen merkkijono lyhempi kuin toinen merkkijono, ovatko merkkijonot samanpituiset tai onko ensimmäinen merkkijono pitempi kuin toinen merkkijono.

   Palauta ohjelma *string_length_comparator.py*-nimisessä ohjelmassa.

   Esimerkki ohjelman toiminnasta, kun syötteet ovat "summer" ja "you think":

```
Hello! I compare the lengths of two strings.
Please, enter the first string:
summer
Please, enter the second string:
you think
"summer" is shorter than "you think".
```

   Esimerkki ohjelman toiminnasta, kun syötteet ovat "pain" ja "life":

```
Hello! I compare the lengths of two strings.
Please, enter the first string:
pain
Please, enter the second string:
life
"pain" is as long as "life".
```

Esimerkki ohjelman toiminnasta, kun syötteet ovat "metre" ja "yard":

```
Hello! I compare the lengths of two strings.
Please, enter the first string:
metre
Please, enter the second string:
yard
"metre" is longer than "yard".
```

5. Kirjoita Python-ohjelma, joka lukee merkkijonon ja kaksi indeksiarvoa ja tutkii ovatko annetuissa paikoissa olevat merkit samat. Pienet ja suuret kirjaimet katsotaan eri merkeiksi. Ohjelma tulostaa virheilmoituksen, jos jompikumpi indeksiarvo tai molemmat indeksiarvot ovat virheelliset. Jälkimmäinen indeksiarvo luetaan, vaikka ensimmäinen indeksiarvo olisikin virheellinen. Laillinen indeksiarvo on välillä [-*n*, *n* - 1], missä *n* on merkkijonon pituus.

   Palauta ohjelmasi *character_comparator.py*-nimisessä tiedostossa.

   Esimerkki ohjelman toiminnasta, kun syötteet ovat "Java", 1 ja 3:

   ```
   Hello! I compare two characters of a string.
   Please, enter a string:
   Java
   Please, enter the first position:
   1
   Please, enter the second position:
   3
   "a" is equal to "a".
   ```

   Esimerkki ohjelman toiminnasta, kun syötteet ovat "Python", -1 ja 0:

   ```
   Hello! I compare two characters of a string.
   Please, enter a string:
   Python
   Please, enter the first position:
   -1
   Please, enter the second position:
   0
   "n" is different from "P".
   ```

   Esimerkki ohjelman toiminnasta, kun syötteet ovat "Tis but a scratch", -18 ja 4:

   ```
   Hello! I compare two characters of a string.
   Please, enter a string:
   Tis but a scratch
   Please, enter the first position:
   -18
   Please, enter the second position:
   4
   Error!
   ```

6. Tee Pythonilla ohjelma, joka lukee merkkijonon ja tulostaa alla annettujen esimerkkien mukaisesti kolmion, joka koostuu kaikista merkkijonon alusta alkavista osajonoista. Voit olettaa, että syötteessä on aina vähintään yksi merkki. Ohjelma tulostaa virheilmoituksen, jos syötteessä on yksi tai useampi välilyönti. Käytä ohjelmassa `for`-silmukkaa. Älä käytä mitään muita kuin luentomateriaalissa mainittuja funktioita. Palauta ohjelmasi *string_triangle.py*-nimisessä tiedostossa. Vinkki: `range`-funktio ja viipalointi.

   Esimerkki ohjelman toiminnasta, kun syöte on "Python":

   ```
   Hello! I print a triangle made of prefixes of a string.
   Please, enter a string:
   Python
   P
   Py
   Pyt
   Pyth
   Pytho
   Python
   ```

   Esimerkki ohjelman toiminnasta, kun syöte on "ooo":

   ```
   Hello! I print a triangle made of prefixes of a string.
   Please, enter a string:
   ooo
   o
   oo
   ooo
   ```

   Esimerkki ohjelman toiminnasta, kun syöte on "x":

   ```
   Hello! I print a triangle made of prefixes of a string.
   Please, enter a string:
   x
   x
   ```

   Esimerkki ohjelman toiminnasta, kun syöte on "this is a test":

   ```
   Hello! I print a triangle made of prefixes of a string.
   Please, enter a string:
   this is a test
   Error!
   ```

7. Kirjoita Pythonilla ohjelman, joka jakaa lukemansa n-merkin mittaisen merkkijonon peräkkäisille riveille. Myös rivien pituus m luetaan käyttäjältä. Kaikilla riveillä on m-merkkiä, jos m jakaa n:n tasan. Muussa tapauksessa viimeinen rivi on vajaa. Tällä rivillä on n % m merkkiä. Viimeinen rivi on samalla ensimmäinen rivi, jos $n \leq m$. Ohjelma tulostaa virheilmoituksen, jos merkkijonossa on yksi tai useampi välilyönti tai $m < 1$. Älä käytä mitään muita kuin luentomateriaalissa mainittuja funktioita. Palauta ohjelmasi *string_wrapper.py*-nimisessä tiedostossa.

   Vinkki: `range`-funktio ja viipalointi.

Esimerkki ohjelman toiminnasta, kun syötteet ovat "Python" ja 2:

```
Hello! I wrap strings into lines.
Please, enter a string:
Python
Please, enter line length:
2
Py
th
on
```

Esimerkki ohjelman toiminnasta, kun syötteet ovat "anaconda" ja 3:

```
Hello! I wrap strings into lines.
Please, enter a string:
anaconda
Please, enter line length:
3
ana
con
da
```

Esimerkki ohjelman toiminnasta, kun syötteet ovat "asp" ja 1:

```
Hello! I wrap strings into lines.
Please, enter a string:
asp
Please, enter line length:
1
a
s
p
```

Esimerkki ohjelman toiminnasta, kun syötteet ovat "Java" ja 4:

```
Hello! I wrap strings into lines.
Please, enter a string:
Java
Please, enter line length:
4
Java
```

Esimerkki ohjelman toiminnasta, kun syötteet ovat "Monty" ja 10:

```
Hello! I wrap strings into lines.
Please, enter a string:
Monty
Please, enter line length:
10
Monty
```

Esimerkki ohjelman toiminnasta, kun syötteet ovat "this is a test" ja 6:

```
Hello! I wrap strings into lines.
Please, enter a string:
this is a test
Please, enter line length:
6
Error!
```

Esimerkki ohjelman toiminnasta, kun syötteet ovat "ni" ja 0:

```
Hello! I wrap strings into lines.
Please, enter a string:
ni
Please, enter line length:
0
Error!
```

Esimerkki ohjelman toiminnasta, kun syötteet ovat "ni" ja -1:

```
Hello! I wrap strings into lines.
Please, enter a string:
ni
Please, enter line length:
-1
Error!
```