1. Seuraavassa esitetään neljä Python-ohjelmaa. Kirjoita kunkin ohjelman viereen sen tuottama tuloste (kukin kohta 3 p., yht. 12 p.).

Ohjelma 1: / Program 1:

```
n = 100
output = "X"

# ulompi if-else rakenne:
#   if-ehto on totta -> sen lohkon (sisennetty) koodi suoritetaan
#   -> ehdon else-haaraa ei suoriteta
if n >= 10:
    print(output)    # tulostuu: X

    # sisempi if-else rakenne:
    #   if-ehto ei ei ole totta (n=100)  -> sen lohkoa ei suoriteta
    #   -> ehdon else-osa suoritetaan
    if n > 150:
        output = "Y"
    else:
        output = "Z"

# tätä ulointa else-osaa ei suoriteta
else:
    print("E")

# tämä tulostuslause suoritetaan aina
print(output)        # tulostuu: Z
```

Tuloste / Output:

```
X
Z
```

Ohjelma 2: / Program 2:

```
i = 1

# toistetaan niin kauan, kun i ei ole
# jaollinen 5:llä, eli jakojäännös ei ole nolla
#  -> kun i on tai 3, niin while ehto toteutuu
#  -> while-toistossa tulostetaan arvot 1 ja 3.
while i % 5 != 0:
    print (i)
    i = i + 2

# kun i saa while-toiston lopussa arvon 5
#   -> while-ehto ei ole totta ->
#   -> lopetetaan while-toisto.
# mutta muuttujan i arvo on edelleen 5
#   -> alla oleva print-lause tulostaa arvon 5
print(i)
```

Tuloste: / Output:

```
1
3
5
```

Ohjelma 3: / Program 3:
 	
`# range on määritelty 3 parametrilla:`
`# alkuarvo = 6, loppuarvo = 1, askel eli muutos = -2`
`# Alkuarvo 6 kuuluu sallittuun alueeseen, loppuarvo 1 ei.`
`# Muuttuja num saa kaikki rangen sallimat arvot`
`#  -> muuttuja num saa arvot 6, 4, 2`
`for num in range(6, 1, -2):`
`  print(f"{num}. item")`
	Tuloste: / Output:
	
	`6. item`
	`4. item`
	`2. item`

Ohjelma 4: / Program 4:
 	
`# monikko ’people’ saa 4 alkiota`
`people = ("Maria", "Ahmed", "Donald", "Olga")`
`# p1 = monikon eka alkio eli ”Maria”, p2 = ”Ahmed” jne.`
`p1, p2, p3, p4 = people`
 	
`# tulostaa: Ahmed (ks. edellä muuttujan p2 arvo)`
`print(p2)`
 	
`# viitataan monikon alkioon indeksillä 2, se onkin`
`# monikon 3.alkio. Indeksointi alkaa nollasta`
`print(people[2])  # tulostaa: Donald`
 	
`# jos p1 eli ”Maria” ei ole (not in) monikossa people`
`#  -> tulostetaan kaikki monikon alkiot.`
`# if-ehto ei ole totta -> ei tulosteta mitään.`
`if p1 not in people:`
`  print(people)`
	Tuloste: / Output:
	
	`Ahmed`
	`Donald`

2. Kirjoita alla olevaan ohjelmaan puuttuvat kaksi funktiota siten, että ohjelma toimii tulostelaatikossa esitetyn esimerkin mukaisesti.

Ohjelma / Program:

```
# Koodaa tähän / Code here:

# Funktio saa 1 parametrin (alkuarvon), käyttäjän antaman luvun
# Funktio palauttaa saamansa luvun 3-kertaisena
def calculate(value):
    '''
    Pidempi varsio:
    tulos = 3 * value       # 'tulos' saa 3-kertaisen arvon.
    return tulos            # palautetaan saatu tulos
    '''

    # pikakoodaus: lasketaan aluksi lauseke (3 * value) ja
    # palautetaan saatu tulos saman tien
    return 3 * value



# Funktio saa 2 parametria:
#   values sisältää listan kokonaislukuja (pääohjelman kutsussa 'list')
#   limits sisältää 2 arvoa, pienimmän ja suurimman sallitun luvun
#  (pääohjelma kutsussa: 'borders')
# Funktio palauttaa listassa olevien niiden lukujen summan, jotka ovat sallitujen
# rajojen sisällä.
def operate(values, limits):
    sum = 0                 # hyväksyttyjen lukujen summa on aluksi nolla.
    min = limits[0]         # pienin luku, joka hyväksytään summan laskentaan
    max = limits[1]         # suurin hyväksytty luku.
    # for-in rakenne käy yksitellen läpi kaikki listan 'values' arvot.
    # listasta saatu arvo sijoitetaan muuttujan 'nr' arvoksi
    for nr in values:
        # jos listasta löytyi sallittujen rajojen sisällä oleva arvo, niin se
        # lisätään laskettavaan summaan.
        if min <= nr <= max:
            sum += nr
    # palautetaan laskettu sallittujen lukujen summa
    return sum



# ---
# Tulostetaan käyttäjän antaman arvo 3-kertaisena.
# Print the value given by the user tripled.
number = int(input("Syötä kokonaisluku / Enter an integer: "))
result = calculate(number)
print(f"Lukusi 3-kertaisena / Your integer tripled: {result}")

# Tulostetaan niiden listassa olevien lukujen summa, jotka ovat sallittujen
# arvojen (nyt 5...15, päätepisteet mukaan lukien) sisällä. Ohjelman täytyy toimia oikein myös muilla
# sallituilla arvoilla.
list = [15, 1, 8, 18, 33]
borders = (5, 15)
result = operate(list, borders)
print(f"Sallittujen lukujen summa / sum of tolerated integers: {result}")
```

Tuloste / Output:

```
Syötä kokonaisluku / Enter an integer: 4
Lukusi 3-kertaisena / Your integer tripled: 12
Sallittujen lukujen summa / sum of tolerated integers: 23
```

3. Seuraavassa on Python-ohjelma, joka käyttää relaatiotietokantaa onnistuneesti. Ohjelman alapuolella on yhdeksän väitettä. Ota kunkin väitteen osalta kantaa siihen, onko väite tosi vai epätosi. (6 oikein = 1 p., 7 oikein = 2 p., 8 oikein = 3 p., 9 oikein = 4 p.).

```
import mysql.connector
1 usage
def fetchEmployees():
    sql = "SELECT id, lastname, firstname, salary FROM person"
    dbcursor = dbconn.cursor()
    dbcursor.execute(sql)
    result = dbcursor.fetchall()
    if dbcursor.rowcount>0:
        for row in result:
            print(f"Salary of {row[2]} {row[1]}: {row[3]} euros per month.")
    return

dbconn = mysql.connector.connect(
         host='127.0.0.1',
         port=3306,
         database='jengi',
         user='apps23',
         password='sAlasana_',
         autocommit=True
         )

fetchEmployees()
```

Oikein	Väärin	Väite	Selitys
	X	Ohjelman suoritus saa tietokantapalvelinohjelmiston käynnistymään.	Ei. Tietokantapalvelinohjelmisto (esim. MariaDB) täytyy olla käynnissä, jotta python-ohjelma voi saada siihen yhteyden.
X		Tietokantapalvelinohjelmisto sijaitsee samassa tietokoneessa kuin Python-ohjelma.	host-arvo (127.0.0.1) viittaa aina omaan koneeseen. Synonyymi numerosarjalle on localhost.
X		Ohjelma muodostaa täsmälleen yhden tietokantayhteyden.	Ohjelma muodostaa 1 kpl tietokantayhteyksiä, sen nimenä on ’dbconn’.
	X	Ohjelma tulostaa toisen, ensimmäisen ja kolmannen henkilön palkan (**salary**) tässä järjestyksessä.	Ei. Select-lause hakee tietokannasta jokaisesta henkilöstä 4 arvoa. for-toiston row sisältää aina yhden henkilön kaikki haetut tiedot. For-toiston print-lauseessa tulostetaan jokaisesta henkilöstä vain sarakkeet (kentät) ’firstname’ eli row[2], ’lastname’ eli row[1] ja ’salary’ eli row[3].
	X	Ohjelma saa hakea tietokannasta korkeintaan 3306 tietuetta.	Ei. Numero 3306 viittaa tietoliikenneportin numeroon, jonka kautta python-ohjelma on yhteydessä tietokantaan. dbconn-asetuksissa: port = 3306.
X		Ohjelman ajo saa tietokantapalvelimen suorittamaan **SELECT**-lauseen yhden kerran.	Kyllä. Pääohjelmassa muodostetaan aluksi tietokantayhteys, muuttuja dnconn. Sen jälkeen pääohjelmassa kutsutaan kerran funktiota fetchEmployees(). Funktiossa suoritetaan vain 1 kerran select-lause ja tulostetaan select-lauseella saatuja arvoja.
	X	Ohjelman tuottamasta tulosteesta nähdään henkilöiden tunnisteet (**id**).	Ei. id-arvo kyllä haetaan tietokannasta, mutta sitä ei tulosteta. Tulostus tapahtuisi funktion for-toiston sisällä olevalla print-lauseella arvolla row[0].
	X	Ohjelma päivittää tietokannassa olevia henkilöiden tunnisteita (**id**).	Ei. Funktiossa on vain 1 kpl select-lauseita, joka pelkästään hakee tietoja.
	X	Tietokantataulu nimeltä **person** ei saa olla tyhjä.	Ei. Jos select-hakulausen ei löydä mitään (taulu on tyhjä), niin koodissa on testi ’if dbcursor.rowcount > 0:’. Saatuja tuloksia yritetään tulostaa vain, jos saatiin oikeasti dataa. Jos tietokannan taulu on tyhjä, niin ei tulosteta mitään.