Datu egiturak

Orain arte datu sinpleak erabili ditugu. Zenbaki bat, testu bat eta abar duten aldagaiekin jolasten aritu gara. Baina badira datu konplexuagoak sortzeko aukera ematen diguten beste datu mota batzuk ere: datu bakar bat baino zerbait gehiago eduki dezakete.

Ordenagailu programek oso gauza zailak egin ditzakete, baina, funtsean, datuak prozesatzen dituzte. Eta askotan, datu horiek sekuentzia luzeetan datoz. Jarraian, datu mota horietako batzuk ikusiko ditugu.

Zerrendak

Zerrendak zenbakiz indexatutako datu-multzo bat dira. Hori definizio oso formala da, baina izenak berak esaten dizu zer diren: zerrenda bat! Datu motei buruzko kapituluan zerrendak aurkeztu genituen eta nola sortzen diren ikusi genuen:

```
hizkuntzak = ["Ingelesa", "Gaztelania", "Frantsesa"]
```

Gogoratu, zenbaki edo indize baten bidez eskura ditzakezula elementuak:

```
hizkuntzak = ["Ingelesa", "Gaztelania", "Frantsesa"]
print (hizkuntzak [2]) #Frantsesa
```

Zerrenda honela errepresentatu daiteke:

0	1	2
"Ingelesa"	"Gaztelania"	"Frantsesa"

3.0 ariketa

Definitu izenen zerrenda bat eta erakutsi pantaila bidez:

```
izenak = ["Ada", "Bug", "Neko"]

print (izenak) # ["Ada", "Bug", "Neko"]
```

Emaitza:

```
["Ada", "Bug", "Neko"]
```

Zerrenden luzera

Zerrenda baten luzera jakiteko, `len()` funtzioa erabil daiteke:

```
hizkuntzak = ["Ingelesa", "Gaztelania", "Frantsesa"]
print (len (hizkuntzak)) # 3
```

3.1 Ariketa

Sortu programa bat 5 zenbaki hamartarrekin definitzeko. Gero, sortu begizta bat zenbaki guztien batez bestekoa kalkulatzeko.

```
zenbakiak = [3.4, 2.7, 4.3, 6.6, 8.3]
batuketa = 0.0

for zenbakia in zenbakiak:
    batuketa = batuketa + zenbakia

batazbestekoa = batuketa / len(zenbakiak)

print("Batazbestekoa da: ", batazbestekoa)
```

Emaitza:

```
Batazbestekoa da:  5.0600000000000005
```

Zerrendatik zatiak atera

Zenbakizko indizea erabiliz, zerrenda baten zatiak atera daitezke, zerrenda horren azpi-zerrenda bat sortuz. Adibidez, "zerrendako lehen hiru balioak nahi ditut", "4. zenbakitik 6.era" edo "azken biak nahi ditut". Horretarako, nahikoa da indize-tarte bat adieraztea:

```
zenbakiak = [1, 2, 3, 4, 5, 6, 7, 8, 9]
zenbakiak [0:4] # [1, 2, 3, 4]
zenbakiak [5:8] # [6, 7, 8]
```

Lehen elementuak ere atera daitezke:

```
zenbakiak = [1, 2, 3, 4, 5, 6, 7, 8, 9]
zenbakiak[: 6] # [1, 2, 3, 4, 5, 6]
```

Edo azken balioak

```
zenbakiak = [1, 2, 3, 4, 5, 6, 7, 8, 9]
zenbakiak[-4:] # [6, 7, 8, 9] =
```

Edo, besterik gabe, azkena:

```
zenbakiak = [1, 2, 3, 4, 5, 6, 7, 8, 9]
zenbakiak[-1]  # [9]
```

Elementuak gehitu eta ezabatu

Zerrenda bati elementu bat gehitu nahi badiogu, nahikoa da `append` funtzioa erabiltzea:

```
hizkuntzak = ["Ingelesa", "Gaztelania", "Frantsesa"]
hizkuntzak.append ("Italiera")
print (hizkuntzak) # ["Ingelesa", "Gaztelania", "Frantsesa", "Italiera"]
```

Eta elementu bat zerrendatik kendu nahi badugu, `del` agindua erabil daiteke:

```
hizkuntzak = ["Ingelesa", "Gaztelania", "Frantsesa"]
del hizkuntzak[1]
print (hizkuntzak) # ["Ingelesa", "Frantsesa"]
```

Azken elementua kentzeko, `pop` funtzioa ere erabil daiteke. Honek elementua kentzen du eta itzultzen du:

```
hizkuntzak = ["Ingelesa", "Gaztelania", "Frantsesa"]
azkena = hizkuntzak.pop()
print(hizkuntzak) # ["Ingelesa", "Gaztelania"]
print(azkena) # "Frantsesa"
```

Eta zerrendako elementu baten balioa ere alda daiteke:

```
hizkuntzak = ["Ingelesa", "Gaztelania", "Frantsesa"]
hizkuntzak[2] = "Italiera"
```

Eta gogoratu, zerrendan zehar joateko, begizta hau erabil dezakegu:

```
hizkuntzak = ["Ingelesa", "Gaztelania", "Frantsesa"]
for hizkuntza in hizkuntzak:
    print (hizkuntza)
```

Indizea erabiliz ere zeharkatu daiteke zerrenda. Horretarako, `range` funtzioa erabili beharko dugu, eta zerrendaren luzera eman beharko diogu, `len` erabiliz:

```
hizkuntzak = ["Ingelesa", "Gaztelania", "Frantsesa"]
for i in range (len (hizkuntzak)):
  print (hizkuntzak [i])
```

Dena den, indizea begiztaren barruan behar ez bada, hobe da zerrenda indizerik gabe egitea, aurreko adibidean egiten den bezala.

Beste lengoai batzuetan, zerrendei `array` esaten zaie. Testuan zehar hitz desberdinak erabiliko ditugu.

3.2 Ariketa

Definitu izenen zerrenda bat eta erakutsi pantaila bidez. Gehitu elementu bat eta erakutsi zerrenda osoa pantailan. Ondoren, zerrendatik elementu bat ezabatu eta berriro pantailan erakutsi zerrenda.

```
izenak = ["Ada", "Bug", "Neko"]

print(izenak) # ["Ada", "Bug", "Neko"]

izenak.append("Miranda")

print(izenak) # ["Ada", "Bug", "Neko", "Miranda"]

del izenak[1]

print(izenak) # ["Ada", "Neko", "Miranda"]
```

Emaitza:

```
["Ada", "Bug", "Neko"]
["Ada", "Bug", "Neko", "Miranda"]
["Ada", "Neko", "Miranda"]
```

Hiztegiak

Hiztegiak datu-multzoak dira, non elementu bakoitzak gako bat eta balio bat ditu. Beste era batera esanda, zerrenda bat bezalakoak dira, baina 0, 1, 2... zenbaki-indizea izan beharrean, zuk nahi duzun balioa dute.

Adibidez, hiztegi bat defini dezakegu, hainbat pertsonaren adinak biltzen dituena, non izena gakoa den eta adina, balioa:

```
adinak = {"Ada": 14, "Bug": 10, "Neko": 2}
print(adinak ["Ada"]) # 14
```

Hiztegia honela adieraz daiteke:

"Ada"	"Bug"	"Neko"
14	10	2

3.3 Ariketa

Defini ezazu `telefonoak` izeneko hiztegi bat, non lagun pare baten telefonoak gordeko dituzun. Gakoa lagunaren izena izango da, eta balioa, berriz, telefono zenbakia.

```
telefonoak = {"Ada": 666555333, "Bug": 111000111 }

print(telefonoak)

for izena in telefonoak.keys():
    print(izena, telefonoak[izena])
```

Emaitza:

```
{'Ada': 666555333, 'Bug': 111000111}
Ada 666555333
Bug 111000111
```

Elementu berriak gehitzeko, balio berri bat esleitu daiteke. Ezabatzeko, `del` agindua erabiltzen da:

```
adinak = {"Ada": 14, "Bug": 10, "Neko": 2}

print (adinak) # {'Ada': 14, 'Bug': 10, 'Neko': 2}

adinak["Miranda"] = 23;

print(adinak) # {'Ada': 14, 'Bug': 10, 'Neko': 2, 'Miranda': 23}

del adinak["Bug"]

print(adinak) # {'Ada': 14, 'Neko': 2, 'Miranda': 23}

for izena in adinak.keys ():
  print (izena, adinak[izena])
```

3.4 Ariketa

Erabili aurreko ataleko telefonoen hiztegia. Eskatu erabiltzaileari datuak hiztegian beste sarrera bat sartzeko: izena eta telefonoa eskatu beharko dituzu, eta gero hiztegira gehitu.

Amaieran, hiztegiko elementu guztiak erakutsi.

```
telefonoak = {"Ada": 666555333, "Bug": 111000111}

izena = input("Sartu izena: ")
telefonoa = input("Sartu telefono zenbakia: ")

telefonoak[izena] = int(telefonoa)

for izena in telefonoak.keys():
    print(izena, telefonoak[izena])
```

Emaitza:

```
Sartu izen bat: Neko
Sartu zenbaki bat: 333222000
Ada 666555333
Bug 111000111
Neko 333222000
```

Hiztegiko balioak ezabatu ditzakegu, honako funtzio honekin:

```
adinak = {"Ada": 14, "Bug": 10, "Neko": 2}
print (adinak) # {'Ada': 14, 'Bug': 10, 'Neko': 2}
del adinak["Bug"]
print (adinak) # {'Ada': 14, 'Neko': 2}
```

Eta hiztegiko balio guztiak erakutsi nahi baditugu? Ez dago arazorik, baina hiztegiaren gako guztiak itzuliko dizkigun funtzio bat erabili beharko dugu: `keys()`. Honela litzateke:

```
adinak = {"Ada": 14, "Bug": 10, "Neko": 2}
for izena in adinak.keys ():
  print (izena, adinak [izena])
```

Pantailaren arabera:

```
Ada 14
Bug 10
Neko 2
```

3.5 Ariketa

Erabili aurreko ataleko telefonoen hiztegia. Eskatu erabiltzaileari izen bat, eta gero kendu elementu hori hiztegitik. Amaieran, hiztegiko elementu guztiak erakutsi.

```
telefonoak = {"Ada": 666555333, "Bug": 111000111 }

izena = input("Sartu izena: ")

del telefonoak[izena]

for izena in telefonoak.keys():
    print(izena, telefonoak[izena])
```

Emaitza:

```
Sartu izen bat: Bug
Ada 66555333
```

Oharra: Beste hizkuntza batzuetan, hiztegiei `hash`, `hashtable` edo "mapak" esaten zaie.

Datu kabiatuen egiturak

Oinarrizko egiturek, hala nola zerrendek eta hiztegiek, zerrendak eta hiztegiak eduki ditzakete kabiatuta.

Datu egitura habiatuak sor daitezke, behar bezain konplexuak. Demagun lagun baten datuak hiztegi honekin irudikatu nahi dituzula:

```
laguna = {"izena": "Neko", "adina": 2}
```

Eta hainbat lagun izango dituen datu egitura bat sortu nahi baduzu? Kasu horretan, hiztegien zerrenda egin dezakezu:

```
lagunak = [
    {"izena": "Neko", "adina": 2},
    {"izena": "Bug", "adina": 10},
    {"izena": "Ada", "adina": 14}
]
print (lagunak [1]) # {"izena": "Bug", "adina": 10}
print (lagunak [2] ["izena"]) #Ada
```

3.6 Ariketa

Idatzi `bezeroak` izeneko hiztegi-zerrenda bat definitzen duen programa bat. Bezeroaren gakoak hauek lirateke: `izena`, `email` eta `adina`. Programak zerrenda egin behar du eta bezero bakoitzaren izena eta adina erakutsi.

```
bezeroak = [
    {
        "izena": "Juan",
        "email": "jj@terra.com",
        "adina": 39
    },
    {
        "izena": "Pedro",
        "email": "pp@ozu.es",
        "adina": 42
    },
    {
        "izena": "Ana",
        "email": "ana@ole.com",
        "adina": 37
    }
]

for bezeroa in bezeroak:
    print(f"{bezeroa['izena']} {bezeroa['adina']}")
```

Emaitza:

```
Juan 39
Pedro 42
Ana 37
```

Hiztegien hiztegia

Eta joko baten egoera, pertsonaiak eta objektuak bilduko dituen datu-egitura bat nahi baduzu? Hiztegi habiatu bat egin nezake, non gakoa pertsonaiaren izena den:

```
pantaila = {
    "Mario": {"bizitza": 10, "objektuak": ["perretxikoa", "izarra"]},
    "Luigi": {"bizitza": 7, "objektuak": []},
    "Toad": {"bizitza": 15, "objektuak": ["perretxiko"]},
}

print (pantaila ["Luigi"] ["bizitza"]) # 7
```

Baina nola dakit zer egitura mota diseinatu behar dudan? Emango diozun erabileraren arabera. Batzuetan denak ibili beharko dituzu, beste batzuetan elementu jakin bat eskuratu beharko duzu... zure programak eskatzen duenaren arabera, egitura jakin bat diseinatu beharko duzu, zure programa azkarrago joan dadin. Orohar, egitura azkarrenak hiztegiak dira, batez ere elementu zehatzak atzitu nahi dugunean.

Testuak

Testu datu mota, katea edo *string* ere deitua, funtsezkoa da programetan. Horregatik, funtzio laguntzaile asko ditu datu mota horiek errazago erabiltzeko. Jarraian, testuetarako baliagarriak diren zenbait funtzio ikusiko ditugu, baina, aurretik, testuari buruzko zerbait azaltzea komeni da:

Testuak zerrendak dira

Izan ere, ordenagailuarentzat, testu bat zerrenda bat bezalakoa da. Letra zerrenda (edo katea) honela erabil dezakegu:

```
testua = "Kaixo naiz Ada"
testua[1] # "a"
```

Testua zerrenda bat balitz bezala ere ibil dezakegu:

```
testua = "Ada"

for karaktere in testua:
    print (karaktere)
```

Irteera hau izango litzateke:

```
A
d
a
```

Testu baten luzera ere jakin dezakegu, `len()` funtzioarekin:

```
testua = "Neko miauka"
len(testua) # 11
```

Baina zalantzarik izanez gero, interesgarriena da testutik nahi dugun zatia atera dezakegula, hasiera eta amaiera adierazita:

```
testua = "Python gora"
testua [0:6] # "Python"
testua [7:12] # "gora"
```

Lehen karaktereak ere atera daitezke

```
testua = "Python gora"
testua[:6] # "Python"
```

Edo azken karaktereak

```
testua = "Python gora"
testua[-4:] # "gora"
```

Edo, besterik gabe, azkena:

```
testua = "Python gora"
testua[-1] # "a" testua
```

Letra larriak/Letra xeheak

Zenbait funtzio ditugu testua letra larriz edo xehez idazteko:

```
testua = "Irakasle Ada"
testua.upper() #IRAKASLE ADA
testua.lower() #Irakasle Ada
```

`title()` izeneko funtzio bat ere badugu, hitz bakoitza testu baten barruan aldatzen duena, lehen letra larriz jarriz.

```
testua = "hau esaldi bat da"
texto.title() #Hau Esaldi bat da
```

split: testutik zerrendara

`split` funtzio interesgarria da testu bat zatika banatzeko eta zerrendan bihurtzeko:

```
testua = "zatoz nirekin, bizi nahi baduzu"
hitzak = testua.split() # ["zatoz", "nirekin", "bai", "nahi duzu", "bizi"]
```

Besterik ezean, `split` operazioa testuaren hutsuneetan mozten da. Baina beste edozein bereizle adieraz daiteke, adibidez:

```
testua = "Ada; Neko; Bug"
izenak = testua.split(";") # [Ada "," Neko "," Bug "]
```

Bilaketak

Batzuetan, testu baten barruan hitz bat bilatu behar dugu. Horretarako, `find` funtzioa erabil dezakegu. Hitza aurkituz gero, hitz hori zein posiziotan hasten den erakutsiko du. Aurkitzen ez badu, `-1` itzuliko du.

```
hitzak = "Irakaslerik onena Ada da, zalantzarik gabe"
aurkitu = hitzak.find ("hobeto") # 3
aurkitu = hitzak.find ("Ada") # 22
EzAurkitua = hitzak.find ("xxx") # -1
```

Testu bat nolabait hasten den jakin nahi badugu, `startswith()` erabil daiteke.

```
hitzak = "Python lengoaia ona da"
hasi = hitzak.startswith ("Py") #True
hasi = hitzak.startswith ("es") #False
```

Testu bat modu batean amaitzen den jakin nahi badugu, berriz, `endswith()` erabil daiteke:

```
hitzak = "Python lengoaia ona da"
acaba = hitzak.endswith ("da") #True
acaba = hitzak.startswith ("da") #False
```

Soberakinak ezabatu

Testuak zuriuneekin edo ezabatu nahi ditugun beste karaktere batzuekin has edo buka daitezke, esaterako, hutsune edo lerro jauziekin. Testu batetik soberan dauden zati horiek kentzeko, honako funtzio hauek erabil daitezke.

`lstrip`ekin, testuaren hasieran espazioak ezabatzen dira:

```
testua = "Espazioak ditut"
garbia = testua.lstrip() # "Espazioak ditut"
```

`rstrip` delakoarekin, testuaren hasieran espazioak ezabatzen dira:

```
testua = "Espazioak ditut"
garbia = testua.rstrip() # "Espazioak ditut"
```

Eta `strip` funtzioarekin bi aldeetako espazioak kenduko ditugu:

```
testua = "Espazioak ditut"
garbia = testua.strip() # "Espazioak ditut"
```

Besterik ezean, espazioak kentzen dira, baina kendu nahi dugun edozein testu adieraz dezakegu:

```
testua = "--Testua-gidoiarekin---"
garbia = testua.lstrip ("-") # "Testua-gidoiarekin---"
```

Fitxategi batetik edo kontsolatik testua irakurtzen dugunean ere, lerro jauziak kentzeko `rstrip` funtzioa erabil daiteke:

```
testua = "Honek lerro-jauzi bat du\n"
garbia = testua.rstrip("\n")
```

Proposatutako ariketak

3.0 Ariketa

Idatz ezazu 5 elementuko zerrenda bat hasieratzen duen programa bat:

- 5 zenbaki zerrenda (`0` balioarekin).
- 5 izenen zerrenda.
- 5 boolear balioko beste bat (`True` balioarekin hasitakoak).

```
izenak = ["Frodo", "Sam", "Merrin", "Pippin", "Bilbo"]
logikoak = [True] * 5
zenbakiak = [0] * 5

print(izenak)
print(zenbakiak)
print(logikoak)
```

Emaitza:

```
[Frodo, Sam, Merrin, Pippin, Bilbo]
[0, 0, 0, 0, 0]
[True, True, True, True, True]
```

3.1 Ariketa

Idatzi 10 zenbakiko zerrenda bat definitzen duen programa bat. Ondoren, begizta bat sortu behar duzu, posizio bikoitietan 0 bat sartzeko.

```
zenbakiak = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for i in range(len(zenbakiak)):
    if i % 2 == 0:
        zenbakiak[i] = 0

print(zenbakiak)
```

Emaitza:

```
[0, 2, 0, 4, 0, 6, 0, 8, 0, 10]
```

3.2 Ariketa

Idatzi zerrenda bat kudeatzeko programa bat, erabiltzaileari 4 aukera dituen menu batekin: 1 elementua sartu, 2 ezabatu, 3 erakutsi eta 4 irten. Menua erabiltzaileak 4. aukera sartzen ez duen bitartean erakutsi behar da. 1 aukera aukeratuz gero, edozein balioko `append` egingo da, 2 aukeratuz gero, `pop` egingo da, eta 3 aukeratuz gero, zerrendako edukia erakutsiko da.

```
zenbakiak = []
aukera = -1

while aukera != "4":
    print("Aukeratu")
    print("1. Elementua sartu.")
    print("2. Elementua kendu.")
    print("3. Arraia erakutsi.")
    print("4. Irten.")
    aukera = input("Sartu aukera: ")

    if aukera == "1":
        berria = int(input("Sartu zenbakia: "))
        zenbakiak.append(berria)
    elif aukera == "2":
        zenbakiak.pop()
    elif aukera == "3":
        print(zenbakiak)
    elif aukera == "4":
        print("Agurra")
    else:
        print("Aukera ezezaguna")
```

Emaitza:

```
Aukeratu aukera
1. Elementua sartu.
2. Elementua ateratzea.
3. Erakutsi zerrenda.
4. Irten.
Aukera ezazu: 3
[]
```

3.3 Ariketa

Idatzi erabiltzaileari hitzak eskatu eta esaldi bat eraikitzeko hitzak kateatzen dituen programa bat, erabiltzaileak puntu bat idatzi arte (.). Azkenean programak sortutako esaldia erakutsi beharko du. Erabiltzaileak ez badu ezer idazten, ez da ezer kateatu behar.

```
izenburua = ""
hitza = ""

while hitza != ".":
    hitza = input("Idatzi hitz bat: ")

    if hitza != "." or hitza != "":
        izenburua = izenburua + " " + hitza

print("Sortutako esaldi:", izenburua)
```

Emaitza:

```
Idatzi hitz bat: Kaixo
Idatzi hitz bat: zer
Idatzi hitz bat: moduz?
Idatzi hitz bat:.
Kaixo, zer moduz? .
```

3.4 Ariketa

Idatzi programa bat erabiltzaileari bere izena, jaioterria eta jaioturtea eskatzeko. Ondoren, esaldi bat erakutsi behar duzu informazio guztiarekin, interpolazioa edo kate txantiloiak erabiliz.

```
izen = input("Idatzi zure izena: ")
jaiotegia = input("Idatzi jaiotze lekua: ")
urtea = input("Idatzi jaiotze urtea: ")

mezua = f"{izen} duzu zinen, {jaiotegia}n jaio zinen {urtea} urtean."

print(mezua)
```

Emaitza:

```
Idatzi zure izena: Ada
Idatzi zure jaioterria: Teverga
Idatzi zure jaioturtea: 2006
Ada duzu izena, Tevergan jaio zinen 2006 urtean.
```

3.5 Ariketa

Idatzi erabiltzaileari esaldi bat eskatzen dion programa bat. Gero esaldi horretako hitz bat eskatu behar du. Azkenkik, programak esaldi bera itzuliko du baina aukeratutako hitza letra larriz agertu beharko da:

```
esaldia = input("Idatzi esaldi bat: ")
hitza = input("Idatzi esalditik hitz bat: ")

posizioa = esaldia.index(hitza)

if posizioa != -1:
    hasiera = esaldia[0:posizioa]
    bukaera = esaldia[posizioa + len(hitza):len(esaldia)]
    emaitza = f"{hasiera}{hitza.upper()}{bukaera}"

    print(emaitza)
else:
    print(hitza, "ez dago esaldian.")
```

Emaitza:

```
Idatzi esaldi bat: Zein ona den Ada andereñoa
Idatzi esalditik hitz bat: ona
Zein ONA den Ada andereñoa
```

3.6 Ariketa

Sortu 5 izeneko zerrenda bat definitzen duen programa bat eta, ondoren, erabili begizta bat izenak banan-banan erakusteko.

```
izenak = ["Frodo", "Merrin", "Sam", "Pip", "Bilbo"]

for izena in izenak:
    print(izena)

# alda-eredu bat:
for i in range(len(izenak)):
```

```
    print(izenak[i])
```

Emaitza:

```
Frodo
Merrin
Sam
Pip
Bilbo
```

3.7 Ariketa

Sortu 10 zenbaki osoko zerrenda bat definitzen duen programa bat. Gero, beste begizta bat sortu, elementu bakoitza inkrementatu (+1) eta erakusteko.

```
zenbakiak = [3, 5, -4, 2, 1, 4, 0, 6, 9, 8, 3]

for zenbakia in zenbakiak:
    print(zenbakia)

for i in range(len(zenbakiak)):
    zenbakiak[i] = zenbakiak[i] + 1

for zenbakia in zenbakiak:
    print(zenbakia)


# Gehiketarako alternatiba:
# zenbakiakGehitu = zenbakiak.map( zenbakia => zenbakia + 1 )
```

Emaitza:

```
3
5
-4
2
1
4
0
6
9
8
3
```

3.8 Ariketa

Sortu 10 zenbaki osoko zerrenda bat definitzen duen programa bat. Ondoren, sortu begizta bat zerrendan elementuren bat errepikatuta dagoen bilatzeko. Errepikatutako bat aurkitzea nahikoa da.

```
numeruak = [3, 5, -4, 2, 1, 4, 0, 6, 9, 8, 3]
errepikatua = False
i = 0
j = 0

while i < len(numeruak) and not errepikatua:
    while j < len(numeruak):
        if numeruak[i] == numeruak[j]:
            errepikatua = True
            break
        j = j + 1
    i = i + 1

if errepikatua:
    print("Zenbaki bat errepikatuta dago")
else:
    print("Ez dago zenbaki errepikaturik")
```

Emaitza:

```
Zenbaki bat errepikatuta dago
```

3.9 Ariketa

Sortu 10 zenbaki osorekin hasitako zerrenda bat definituko duen programa. Gero, beste begizta bat sortu, zenbaki positiboak, negatiboak eta 0 direnak zenbatuko dituena.

```
zenbakiak = [3, 5, -4, 2, 1, 4, 0, 6, -9, 8, 3]

positiboak = 0
negatiboak = 0
hutsak = 0

for zenbakia in zenbakiak:
    if zenbakia > 0:
        positiboak = positiboak + 1
    elif zenbakia < 0:
        negatiboak = negatiboak + 1
    else:
        hutsak = hutsak + 1

print("Positiboak: ", positiboak)
print("Negatiboak: ", negatiboak)
print("Hutsak: ", hutsak)
```

Emaitza:

```
Positiboak: 8
Negatiboak: 2
Zeroak: 1
```

3.10 Ariketa

Sortu 5x10 elementuko bi dimentsioko zerrenda bat definituko duen programa bat. Zerrendako balioak ausazko (aleatorio edo *random*) zenbakiak erabiliz hasten dituen begizta bat sortzen du. Zenbaki aleatorioak sortzeko, `random` eta `random.randint()` funtzioa erabiltzen du liburutegiak, hemen erakusten den bezala:

```
import random
random.randint(0, 30); # 0 eta 30 arteko ausazko zenbakia
```

Horren ondoren, sortu beste begizta bat, elementuren batean `15` zenbakia aurkituz gero begizta eten eta zer posiziotan dagoen erakutsiko duena.

```
import random

matrizea = [([0] * 10)] * 5

print(matrizea)

for i in range(len(matrizea)):
    random.seed()
    for j in range(len(matrizea[i])):
        matrizea[i][j] = random.randint(0, 30)

print(matrizea)

for i in range(len(matrizea)):
    for j in range(len(matrizea[i])):
        if matrizea[i][j] == 15:
            print("15 aurkitu da ", i, j)
```

3.11 Ariketa

Sortu 10 zenbaki osorekin hasitako zerrenda bat definitzen duen programa. Begizta batean, pantaila bidez erakusten ditu elementu guztiak. Gero, beste begizta bat sortzen du, eta, bertan, elementuak lekuz aldatzen aldatzen ditu, indizeetan `random` funtzioa erabiliz. Gero emaitza erakusten du.

```
import random

zenbakiak = [4, 7, -3, 7, 1, 11, 9, 0, 5, 8]

print(zenbakiak)

for i in range(len(zenbakiak)):
    ausazko_indizea = random.randint(0, len(zenbakiak) - 1)
    aurrekoa = zenbakiak[i]
    zenbakiak[i] = zenbakiak[ausazko_indizea]
    zenbakiak[ausazko_indizea] = aurrekoa

print(zenbakiak)
```

Emaitza:

```console
[5, 4, 11, 7, 1, -3, 0, 9, 7, 8]
```