2. Oletetaan, että luku x on kiinnostava, jos se koostuu vain ja ainoastaan luvun y numeroista siten, että kukin y:n numero esiintyy x:ssä vähintään kerran. Voit olettaa selkeyden vuoksi, että kukin y:n numeroista esiintyy y:ssä vain kerran. Näin y ei voi olla esimerkiksi 112, koska numero 1 esiintyy kahdesti.

   Oletetaan esimerkiksi, että $y = 103$. Tällöin esimerkiksi luku 33101 on kiinnostava, koska y:n numerot 1, 0 ja 3 esiintyvät luvussa 33101 vähintään kerran eikä luvussa ole muita numeroita kuin 1, 0 ja 3. Toisaalta esimerkiksi luku 33000 ei ole kiinnostava, koska siitä puuttuu numero 1. Luku 33540 ei ole kiinnostava kahdesta syystä: siinä on y:stä puutuvat numerot 5 ja 4 ja siitä puuttuu y:n numerot 1 ja 0.

   Tee Python-ohjelma, joka lukee luvut x ja y käyttäjältä ja kertoo onko x kiinnostava vai ei. Palauta ohjelma *relevant_to_my_interests.py*-nimisessä tiedostossa.

   Vinkki: käsittele lukuja merkkijonoina tai muina sekvensseinä.

   Älä käytä `import`-lauseella käyttöön otettujen Pythonin moduulien funktioita tai metodeja. Älä käytä tehtävässä myöskään `set`- ja `frozenset`-tyyppejä. Ratkaisusta saa nolla pistettä, jos siinä käytetään kiellettyjä Pythonin piirteitä, vaikka ohjelma toimisi oikein.

   Esimerkki ohjelman toiminnasta, kun syötteet ovat 33101 ja 103:

```
Hello! I am interested in numbers.
Please, enter x:
33101
Please, enter y:
103
This is relevant to my interests.
```

   Esimerkki ohjelman toiminnasta, kun syötteet ovat 33000 ja 103:

```
Hello! I am interested in numbers.
Please, enter x:
33000
Please, enter y:
103
This is not relevant to my interests.
```

   Esimerkki ohjelman toiminnasta, kun syötteet ovat 33540 ja 103:

```
Hello! I am interested in numbers.
Please, enter x:
33540
Please, enter y:
103
This is not relevant to my interests.
```

   Esimerkki ohjelman toiminnasta, kun syötteet ovat 1084 ja 1048:

```
Hello! I am interested in numbers.
Please, enter x:
1084
Please, enter y:
1048
This is relevant to my interests.
```

3. Tee Pythonilla sananeliöitä tulostava ohjelma. Neliöitävä sana luetaan käyttäjältä. Voit olettaa, että syötteessä ei ole välilyöntejä. Palauta ohjelma *word_square.py*-nimisessä tiedostossa. Vinkki: saat käännettyä sanan merkkien järjestyksen päinvastaiseksi muun muassa viipaloimalla (`[::-1]`-notaatio) tai `reversed`-funktiolla.

   Esimerkki ohjelman toiminnasta, kun syöte on "cat":

```
Hello! I shall square a word.
Please, enter a word:
cat
Squared:
cat
a a
tac
```

   Esimerkki ohjelman toiminnasta, kun syöte on "worm":

```
Hello! I shall square a word.
Please, enter a word:
worm
Squared:
worm
o  r
r  o
mrow
```

   Esimerkki ohjelman toiminnasta, kun syöte on "python":

```
Hello! I shall square a word.
Please, enter a word:
python
Squared:
python
y     o
t     h
h     t
o     y
nohtyp
```

   Esimerkki ohjelman toiminnasta, kun syöte on "ab":

```
Hello! I shall square a word.
Please, enter a word:
ab
Squared:
ab
ba
```

   Esimerkki ohjelman toiminnasta, kun syöte on "x":

```
Hello! I shall square a word.
Please, enter a word:
x
Squared:
x
```

4. Kirjoita Pythonilla *summaa_väli*-niminen funktio, jossa lasketaan indeksiarvojen väliin kuuluvien kokonaislukujenlistan alkioiden summa. Lista on funktion ensimmäinen parametri. Toinen parametri on indeksiarvo a (kokonaisluku), jonka määrittämässä paikassa on ensimmäinen summaan mukaan laskettava listan alkio. Kolmas parametri on indeksiarvo b (kokonaisluku), jonka määrittämässä paikassa on viimeinen summaan mukaan laskettava listan alkio. Funktio palauttaa paluuarvonaan välillä olevien alkioiden summan. Jos parametrien arvot ovat esimerkiksi [11, 22, 33, 44], 1, 2, niin summaan lasketaan alkiot 22 ja 33 ja paluuarvo on siten 55. Paluuarvo on Pythonin `None`-vakio, jos $a < 0$, $b \geq n$, missä n on listan alkioiden lukumäärä, $a > b$ tai $n = 0$ eli ensimmäisen parametrin arvo on tyhjä lista `[]`. Funktio ei tulosta mitään.

   Luo lista pääohjelmassa ja täytä se käyttäjän antamilla syöteillä siten, että käyttäjältä luetaan ensin lukujen määrä, jonka jälkeen luvut luetaan yksitellen. Lista jää tyhjäksi, kun käyttäjä antaa määräksi nollan tai negatiivisen luvun. Käyttäjältä luetaan lopuksi indeksiarvojen kokonaislukuina välin alaraja (*a*) ja yläraja (*b*). Rajat luetaan, vaikka lista on tyhjäksi.

   Välitä lista ensimmäisenä, alaraja toisena ja yläraja kolmantena parametriarvona tekemällesi funktiolle. Sijoita funktion palauttama arvo pääohjelmassa muuttujaan. Tulosta tulos näytölle pääohjelmassa muuttujan avulla. Palauta ohjelma *element_summer.py*-nimisessä tiedostossa.

   Älä käytä `import`-lauseella käyttöön otettujen Pythonin omien moduulien funktioita tai metodeja. Ratkaisusta saa nolla pistettä, jos siinä käytetään kiellettyjä toimintoja, vaikka ohjelma toimisi oikein.

   Lisää tämän tehtävän ratkaisujen loppuun `if`-lause, jonka avulla ohjelmaa voidaan käyttää sekä skriptinä että moduulina:

```
# Kutsutaan pääohjelmaa, jos ohjelmaa ajetaan.
if __name__ == "__main__":
    main()
```

   Esimerkki ohjelman toiminnasta, kun käyttäjän syötteet ovat 4, 11, 22, 33, 44, 1 ja 2:

```
Hello! I sum elements.
Please, enter the number of integers:
4
Please, enter an integer:
11
Please, enter an integer:
22
Please, enter an integer:
33
Please, enter an integer:
44
Please, enter a lower bound:
1
Please, enter an upper bound:
2
The sum is 55.
```

Esimerkki ohjelman toiminnasta, kun käyttäjän syötteet ovat 3, 10, -7, 3, 1 ja 1:

```
Hello! I sum elements.
Please, enter the number of integers:
3
Please, enter an integer:
10
Please, enter an integer:
-7
Please, enter an integer:
3
Please, enter a lower bound:
1
Please, enter an upper bound:
1
The sum is -7.
```

Esimerkki ohjelman toiminnasta, kun käyttäjän syötteet ovat 1, -2, 0 ja 0:

```
Hello! I sum elements.
Please, enter the number of integers:
1
Please, enter an integer:
-2
Please, enter a lower bound:
0
Please, enter an upper bound:
0
The sum is -2.
```

Esimerkki ohjelman toiminnasta, kun käyttäjän syötteet ovat 2, -2, 42, 1 ja 0:

```
Hello! I sum elements.
Please, enter the number of integers:
0
Please, enter a lower bound:
0
Please, enter an upper bound:
0
Could not compute!
```

Esimerkki ohjelman toiminnasta, kun käyttäjän syötteet ovat 2, 3, 4, 1 ja 10:

```
Hello! I sum elements.
Please, enter the number of integers:
2
Please, enter an integer:
3
Please, enter an integer:
4
Please, enter a lower bound:
1
Please, enter an upper bound:
10
Could not compute!
```