1. Täydennä seuraava koodi toimivaksi. Kirjoita koodia vain harmaisiin kohtiin. Hyödynnä periytymissuhdetta eli älä kirjoita samaa koodia useaan kertaan. Ohjelman tulee toimia samoin kuin esimerkkitulosteessa.

```
class Person:
    def __init__(self, first_name, last_name):
```

```
        self.first_name = first_name;
        self.last_name = last_name;
```

```
    def plot(self):
```

```
        print(f"-- My name is {self.first_name} {self.last_name}");
        return;
```

```
class Student(Person):
    def __init__(self, first_name, last_name, student_nr):
        super().__init__(first_name, last_name);
        self.student_nr = student_nr;


    def plot(self):
        super().plot();
        print(f"and my student number is {self.student_nr}")
```

```
# pääohjelma

p1 = Person("James", "Bond");
s1 = Student("Johnny", "English", 321);
p1.plot();
s1.plot();
```

Ohjelman tuottama tuloste:

```
-- My name is James Bond
-- My name is Johnny English
and my student number is 321
```

2. Määrittele muutamalla lauseella seuraavat olio-ohjelmoinnin käsitteet

a) luokka / class

- luokan avulla voidaan luoda olioita
- määrittelee olioiden yhteiset ominaisuudet ja toiminnot
- ’sapluuna’ olioiden luomiseksi

b) assosiaatio

- kahden luokan välinen yhteys/ suhde
- suhde voi olla pysyvä tai tilapäinen.
- pysyvässä suhteessa luokka tietää/ tuntee toisen luokan koko ajan

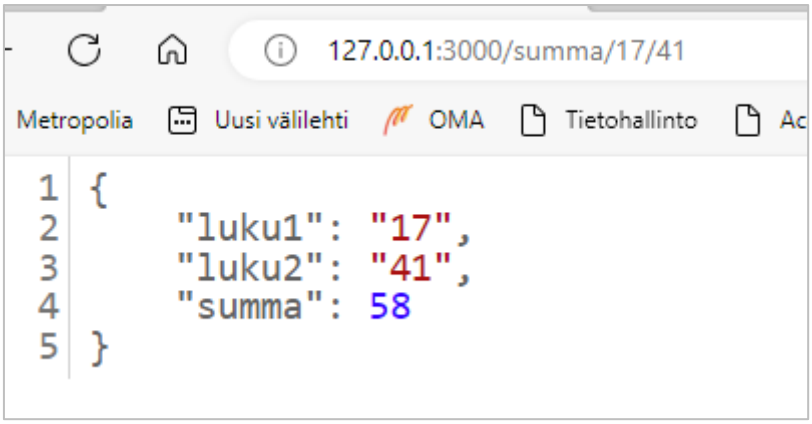
c) päätepiste / endpoint

- määrittää yksittäisen palvelun, jota asiakas voi käyttää
- jokaisella päätepisteellä on oma yksilöllinen URL-osoite
- yhdistää tulevan API-kutsun oikeaan funktioon, joka vastaa siihen.

d) Flask

- pythonin kirjasto / lisäosa
- mahdollistaa taustapalvelun/ ohjelmointirajapinnan (API) rakentamisen
- käynnistää taustapalvelun, joka odottaa kutsuja

3. Alla on kuva selaimesta lähetetystä API-kutsuista ja siihen saadusta vastauksesta. Täydennä ohjelmassa olevat varjostetut kohdat siten, että API tuottaa selaimeen vastaavat tiedot kuin kuvassa. Virheisiin ei tarvitse varautua. (max. 8 p.)




```
from flask import Flask, Response


app = Flask(__name__)
@app.route(' /summa/<luku1>/<luku2> ')
def kaiku( luku1, luku2 ):

    summa = float(luku1) + float(luku2)

    vastaus = {
      "luku1" : luku1,
      "luku2" : luku2,
      "summa" : summa
    }

    return vastaus


if __name__ == '__main__':
    app.run(use_reloader=True, host=' 127.0.0.1 ', port= 3000 )
```