```
class Ucenik:
 def __init__(self, ime_ucenika):
     self.name = ime_ucenika
     self.subjects = {}

 def unesi_ocjenu(self, ocjena, predmet, datum_unosa):
     if predmet not in self.subjects:
        self.subjects[predmet] = []
     self.subjects[predmet].append(
           {"ocjena": ocjena, "datum_unosa": datum_unosa})

 def prosjek_predmet(self, predmet):
     if predmet not in self.subjects:
        return 0
     ukupni_zbroj_ocjena = sum(
          entry["ocjena"] for entry in self.subjects[predmet])
     return ukupni_zbroj_ocjena / len(self.subjects[predmet])

 def predmeti_rank(self):
     lista_predmeta = list(self.subjects.keys())
     lista_predmeta.sort(key=lambda
      predmet: self.prosjek_predmet(predmet), reverse=True)
     return lista_predmeta

def globalni_prosjek(self):
     prosjek_zbroja_predmeta = 0
     broj_predmeta = 0
     broj_svih_ocjena = 0
     zbroj_svih_ocjena = 0

     for predmet, entries in self.subjects.items():
        prosjek_zbroja_predmeta += self.prosjek_predmet(predmet)
        broj_predmeta += 1
        for entry in entries:
           zbroj_svih_ocjena += entry["ocjena"]
           broj_svih_ocjena += 1

        prosjek_po_predmetu = prosjek_zbroja_predmeta
             / broj_predmeta if broj_predmeta > 0 else 0
        prosjek_svih_ocjena = zbroj_svih_ocjena /
            broj_svih_ocjena if broj_svih_ocjena > 0 else 0
         return prosjek_po_predmetu, prosjek_svih_ocjena

multiple = vrijednost_ako_je_true if usporedba
else druga_vrijednost
raise Objekt_Tipa (raise ValueError('number must be non negative')

potenciranje --> **, nad bitovima --> &, | , ^, logicke --> and, or, not
a=a/5  <--> a/=5
```

```
class Razred:
   def __init__(self, ime_razreda, lista_ucenika):
      self.className = ime_razreda
      self.studentsList = lista_ucenika

   def ucenici_rank(self, predmet):
      return sorted(self.studentsList, key=lambda ucenik:
ucenik.prosjek_predmet(predmet), reverse=True)

   def save(self, file_path):
      with open(file_path, "wb") as file:
         pickle.dump(self, file)

   def load(cls, file_path):
      with open(file_path, "rb") as file:
         return pickle.load(file)


class UcenikPlus(Ucenik):
   def __init__(self, ime_ucenika):
      super().__init__(ime_ucenika)

   def prosjek_predmet(self, predmet):
      klasicni_prosjek =  super().prosjek_predmet(predmet)
      zaokruzeni_prosjek = math.ceil(klasicni_prosjek)
      return zaokruzeni_prosjek

   def globalni_prosjek(self):
      p1, p2 = super().globalni_prosjek()
      return math.ceil(p1), math.ceil(p2)
```

spajanje lista
nova = lista1 + lista2

umetanje elementa
nova.insert(1, 543)

broj pojavljivanja
[1,2,3].count(2)

index prvog pojavljivanja
[1,2,3].index(2)

uklanja prvo pojavljivanje vrijednosti
[1,2,3,3,4].remove(3) = [1,2,3,4]
[1,2,3,4].pop() - zadnji ili n-ti element
[1,2,3,4].pop(2) - index koji ide od 0,1,2

duboko kopiranje
promjena se vidi, obje varijbale pokazuju na istu listu
lista1 = [0,1,2], lista2 = lista1

plitko kopiranje - imena pokazuju na razlicite liste
lista1 = [0,1,2], lista2 = lista1[:]

Skupovi --> set - s = {1, (2,3), '4', 5, 5} - skup = {1, 2, 3, 4}
- skup.add(4) - skup.update({4,6}),
skup.discard(4) - uklanja element, nema exceptiona

skup.remove(4) - uklanja element, ima exceptiona ako nema elementa

skup.pop() - baca exception ako je prazan, pop uzima nasumicno odabran