# 1 Objekt posuđuje novac od drugog objekta. Kreiramo klasu osoba, u -te atribute objekta ime u __init__. Nadodati property stanje= 10000 (ne unosi se u konstruktoru ali je definiran u init metodi)

## 1.1 Kreirati metodu repr koja samo ispisuje ime objekta

## 1.2 Definirati funkciju posudba koja prima dva argumenta (iznos novca za posudbu te ime objekta od kojeg se posuđuje novac.
kako je novac posuđen tako se na stanje jednog objekta dodaje novac a sa stanja drugog objekta se oduzima isti iznos

```
class Osoba:
  def __init__(self, ime):
    self.ime = ime
    self.stanje = 10000

  def __repr__(self):
    return self.ime
  def posudba(self, iznos, other):
    self.stanje += iznos
    other.stanje -= iznos

o1 = Osoba( "pero")
o2 = Osoba( "ivo")
print(o2.stanje)
o1.posudba(300,o2)
print(o1.stanje)
print(o2.stanje)
```

nadopunimo zadatak tako da nije moguće posuditi novac od osobe koja nema novca

```
def posudba(self, iznos, other):
    if other.stanj >= iznos:
      self.stanje += iznos
      other.stanje -= iznos
    else:
      print("ništa od posudbe, nemam ni kinte")
```

1 object vs object

## 2 Objekti- student- kreirati klasu student te atribute objekta **ime** u __init__. Nadodati property **groupMember**= None (ne unosi se u konstruktoru ali je definiran u init metodi)

### 2.1 Kreirati repr gdje se ispisuje samo ime

### 2.2 Kreirati metodu setGroupMember koja postavlja kolegu našem objektu

```
class Student:
   def __init__(self, ime):
      self.ime = ime
      self.groupMember = None
   def __repr__(self):
#želimo da se ipsiše ime studenta a ne objekt
      return f"{self.ime} je naziv ovog objekta"



   def setGroupMember(self, other):
      self.groupMember = other
      other.groupMember = self

std1 = Student("Ankica")
std2 = Student("Mojimir")
std1.setGroupMember(std2)
print(std1.groupMember)
print(std2.groupMember)
```

### 2.3 postavljamo uvjet u metdu setGroupMember slučaju da je groupMember već postavljen da se ispisuje poruka, imamo partnera

```
        def setGroupMember(self, other):
            if self.groupMember == None:
               self.groupMember = other
               other.groupMember = self
            else:
               print(self.ime, " već ima kolegu ")
std3 = Student("Kocka")
std4 = Student("Brid")
std2.setGroupMember(std3)
print("----------")
```

2 object vs object

```
print(std3.groupMember)
```

# 3 Zadatak kao prethodni samo da se omogući kreiranje grupe od sve skupa tri člana, koristimo listu gdje spremamo ostala dva člana grube od zadanog objekta

```
class Student:
    def __init__(self, ime):
        self.ime = ime
        self.groupMember = [None, None]

    def __repr__(self):
        # kako želimo
        return f"{self.ime} je član grupe"

    def setGroupMember(self, other, third):
        self.groupMember[0] = other
        self.groupMember[1] = third
        other.groupMember[0] = self
        other.groupMember[1] = third
        third.groupMember[0] = self
        third.groupMember[1] = other

std1 = Student("Ankica")
std2 = Student("Mojimir")
std3 = Student("Onjko")
std1.setGroupMember(std2, std3)
```

## 3.1 Izmijeniti zadatak tako da se grupa sastoji od proizvoljnog broja članova

```
class Student():
    def __init__ (self,ime):
        self.ime=ime
        self.GroupMember=[]
        self.uGrupi=False

    def __repr__(self):
        return f'naziv je {self.ime}'
```

3 object vs object

```
    def setGroupMember(self,*kwargs):
        self.uGrupi=True
        for element in kwargs:
            self.GroupMember.append(element)
            element.uGrupi=True
    # self.GroupMember=[s2,s3,s4]
        for element in kwargs:
            element.GroupMember=self.GroupMember[:] #punimo svakom elementu grupu sa "Ivicinim" članovima grupe
            element.GroupMember.append(self)
            #element.GroupMeber=[s1,s2,s3,s4]
            element.GroupMember.remove(element)
            #brišemo element kako ne bi imao samog sebe u članovima grupe

s1=Student("Ivica")
s2=Student("Marica")
s3=Student("Perica")
s4=Student("Zdeslav")

s1.setGroupMember(s2,s3,s4)
print(s4.GroupMember)
```

# 4 Objekt posuđuje nekakav predmet (item) od drugog objekta. Kreiramo klasu osoba, u - te atribute objekta ime u __init__ i rječnik popis_stvari.

```
o1 = Osoba("ivo", {"kuglica": 4, "kvadratić": 2, "rombić": 0})

o2 = Osoba("ana", {"kuglica": 1, "kvadratić": 3, "valjak": 5})
```

## 4.1 Definirati funkciju posudba koja prima dva argumenta (item- naziv itema koji se posuđuje posudbu te ime objekta od kojeg se posuđuje item.
kako je item posuđen tako se u rječniku skida broj s tog itema odnosno dodaje broj itema onome kojem je posuđen. Ukoliko osoba od koje se posuđuje ima manje od 1 itema ispisuje se nemam tu stvar.

```
class Osoba:
    def __init__(self, ime, popis_stvari):
```

```
        self.ime = ime
        self.popis_stvari = popis_stvari

    def posudba(self, item, other):
        if item in other.popis_stvari.keys() and other.popis_stvari[item] > 0:
            if item in self.popis_stvari.keys():
                self.popis_stvari[item] += 1
            else:
                self.popis_stvari[item] = 1
            other.popis_stvari[item] -= 1
        else:
            print("nemam tu stvar")
o2.posudba("rombić", o1)

o2.posudba("kuglica", o1)

print(o2.popis_stvari)

print(o1.popis_stvari)
```

# 5 Kreirati klasu Kosarka koja se sastoji od argumenta naziv kluba, unutar konstruktora. Unutar init metode kreirati još dva argumenta: self.postigao i self.primio, postaviti im vrijednosti na 0

## 5.1 Ispisati koliko koševa je zabio pojedini klub kao domaćin(metoda objekta)

## 5.2 Ispisati sve ukupno koliko su domaćini ukupno primili koševa po utakmici (za sve klubove domaćine prosjek primljenih koševa)

## 5.3 Ispisati sve susrete zajedno s rezultatima kao na slici:

5.4 150 koliko je zabio klub kao domaćin cibona

5.5 66.66666666666667 prosječno primljenih po utakmici

5.6 cibona-zadar 70:100

5.7 cibona-Olimpija 80:50

5.8 Šibenik-cibona 80:50

```
class Kosarka:
    br_susreta = 0
    ukupni_primljeni = 0
    svi_susreti = {}
```

5 object vs object

```
    def __init__(self, naziv_kluba):
        self.naziv_kluba = naziv_kluba
        self.ukupno_zabijeni_klub = 0
    def susret(self, postigao, primio, other):
        self.postigao = postigao
        self.primio = primio
        other.postigao = primio
        other.primio = postigao
        self.ukupno_zabijeni_klub += postigao
        Kosarka.br_susreta += 1
        Kosarka.ukupni_primljeni += primio
        key = self.naziv_kluba+"-"+other.naziv_kluba + " "
        value = str(postigao)+":"+str(primio)
        Kosarka.svi_susreti[key] = value
    @classmethod
    def ispis_susreta(cls):
        for k, v in Kosarka.svi_susreti.items():
            print(k, v)


k1 = Kosarka("cibona")
k2 = Kosarka("zadar")
k3 = Kosarka("Olimpija")
k4 = Kosarka("Šibenik")
k1.susret(70, 100, k2)
k1.susret(80, 50, k3)
k4.susret(80, 50, k1)
print(k1.ukupno_zabijeni_klub, " koliko je zabio klub kao domaćin", k1.naziv_kluba)
print(Kosarka.ukupni_primljeni/Kosarka.br_susreta,
      " prosječno primljenih po utakmici")
Kosarka.ispis_susreta()
```

6 object vs object

## 6 Kreirati klasu razlomak, u konstruktor se unose brojnik i nazivnik (b,n) te se kreira metodu umnožak koja vraća umnožak dvaju razlomaka(instanci- jer je svaki razlomak jedan objekt). Rezultat također treba biti razlomak (novi objekt).

```
class razlomak:
   def __init__(self,b,n):
      self.b=b
      self.n=n
   #r podrazumjeva umetnunti objekt u metodu umnozak
   def umnozak(self,r):
      b=self.b*r.b
      n=self.n*r.n
      t=razlomak(b,n)
      return t
a=razlomak(5,6)
b=razlomak(3,10)
c=a.umnozak(b)
print(c.b)
print(c)
```

## 7 (kolokvijalno)prethodnom zadatku ćemo nadodati metodu podrezi i umetnuti ju u metodu umnozak a i u __init__metodu kako bi se sam razlomak na početku skratio.

```
class razlomak:
   def __init__(self, b, n):
      self.b = b
      self.n = n

   def umnozak(self, other):
      brojnik = self.b*other.b
      nazivnik = self.n*other.n
      t = razlomak(brojnik, nazivnik)
      return t

   def __repr__(self):
```

```
        return f"{self.b}/{self.n}"


r1 = razlomak(4, 8)
r2 = razlomak(5, 2)
novi = r1.umnozak(r2)
print(r1)
print(novi)
```

## 7.1 Ukoliko želimo nadodati funkciju gdje se razlomak sam skraćuje kreiramo funkciju podrezi (nastavak na zadatak razlomak) i implementiramo ju u funkcije umnozak i __repr__

```
    def podrezi(self):
        i = 2
        b = self.b
        n = self.n
        while i < b and i < n:
            while b % i == 0 and n % i == 0:
                n = n//i
                b = b//i
            i += 1
        self.b = b
        self.n = n
        t = razlomak(self.b, self.n)
        return t
    def umnozak(self, other):
        brojnik = self.b*other.b
        nazivnik = self.n*other.n
        t = razlomak(brojnik, nazivnik)
        t = t.podrezi()
        return t

    def __repr__(self):
        self = self.podrezi()
        return f"{self.b}/{self.n}"
```

8 object vs object

Kolokvijalno Dodajmo ugrađenu metodu za ispis __str__(self):

```
  def __str__(self):
     if self.n==1:
        return ('{}'.format(self.b))
     else:
        return ('{}/{}'.format(self.b,self.n))
```

Umjesto funkcije umnožak koristimo definiranu metodu __mul__(self,r):

```
  def __mul__(self,r):
     b=self.b*r.b
     n=self.n*r.n
     return razlomak(b,n)
```

Dodajmo metode za zbrajanje, oduzimanje i dijeljenje

```
def __add__(self,r):
          n=self.n*r.n
          b=self.b*r.n+self.n*r.b
          return razlomak(b,n)
     def __sub__(self,r):
          n=self.n*r.n
          b=self.b*r.n-self.n*r.b
          return razlomak(b,n)
     def __truediv__(self,r):
          n=self.n*r.b
          b=self.b*r.n
          return razlomak(b,n)


a=razlomak(4,5)
b=razlomak(7,6)
print(a, " a")
print(b, " b")
print(a*b, " a*b")
print(a+b," a+b")
print(a-b," a-b")
print(a/b," a/b")
```