Domaći zadatak 3

1. Napisati funkciju koja za zadati string i slovo vraća listu torki. Svaka torka se odnosi na rečenicu, a sastoji se od svih riječi koje se završavaju sa zadatim slovom u toj rečenici, kao i broj riječi koje se završavaju sa zadatim slovom u toj rečenici. Primjer: get_words_ends_with_letter (“Print only the words that end with the chosen letter in those sentences. Example can contains one or more sentences.”, “s”) vraća listu sljedećeg oblika: [(“words”, “sentences”, 2), (“contains”, “sentences”, 2)]

2. Napisati program koji nalazi ponavljajuću sekvencu (podniz) koja u proizvodu daje najveću vrijednost. Štampati tu sekvencu i proizvod te sekvence. Primjer: za listu [1, 2, 2, 2, 4, 4] treba da se štampa sekvenca [4, 4] i proizvod 16.

3. Napisati Python funkciju kojom tražite najduži testerasti podniz. Testerasti podniz je onaj za koji važi da je prvi broj manji od drugog, drugi da je veći od trećeg, treći manji od četvrtog, itd.

4. Za zadati string napisati program kojim se provjerava koliko puta se u stringu *dva susjedna* karaktera ponavljaju (*poklapaju*).
   Primjer: aabaaac rezultat je 3.

5. Za svaki *podcast* je poznat naziv, broj pozitivnih komenatara na *podcastu* i broj negativnih komentara na *podcastu*. Vaš zadatak je da kreirate program kojim pronalazite *podcast* koji ima najgori odnos između pozitivnih i negativnih komentara.
   Primjer: Za niz
   [{‘naziv’:’Español para principiantes’, ‘br_pozitivni’:1000,’br_negativni’:10},
   {‘naziv’:’Philophize This!’, ‘br_pozitivni’:500,’br_negativni’: 30}, {‘naziv’:’Science VS. ’,
   ‘br_pozitivni’:600,’br_negativni’: 45}]
   treba da štampa *Science VS*.

6. Napisati klasu **Televizor** koja se sastoji od sledećih atributa: broj tekućeg kanala (int, između 0 i dužine od liste kanala), naziv tekućeg kanala (string), kanali (lista stringova), jačina tona (int, između 0 i 10). Potrebno je odraditi sledeće:
    a. Napisati konstruktor kojim se postavljaju početne vrijednosti atributa. Za atribut kanali to treba da bude prazna lista.
    b. Kreirati odgovarajući **getere** i **setere**. Obratiti pažnju na ograničenja za attribute broj kanala i jačina tona.
    c. Napisati funkciju **dodaj_kanal** sa parametrom naziv kanala, koja dodaje novi kanal u listu kanala.
    d. Napisati funkciju **obriši_kanal** sa parametrom na naziv kanala, kojom se briše kanal na osnovu naziv.
    e. Napisati funkciju **pojačaj_ton** koja uvećava vrijednost tona za jedan u odnosu na trenutnu jačinu
    f. Napisati funkciju **ime_kanala** sa parametrom za broj kanala, koja vraća naziv kanala na osnosu zadatog broja.Prvi kanal se nalazi na poziciji 1.

7. Potrebno je implementirati jednostavnu biblioteku za upravljanje knjigama u programskom jeziku Python. **Biblioteka** treba omogućiti dodavanje, pregled, uređivanje i brisanje knjiga iz inventara. Potrebno je kreirati klasu *Book* i klasu *Library*.
    - Svaka knjiga ima sledeće atribute: *naslov*, *autor*, *godina izdanja* i *broj kopija u inventaru*.
    - Implementirati konstruktor klase *Book* koji će inicijalizovati atribute knjige. Implementirati metode za dohvatanje i postavljanje svakog atributa knjige (geteri i seteri).
    - *Biblioteka (Library)* treba da sadrži listu knjiga koje su dostupne u inventaru. Implementirati metode za dodavanje knjige u inventar (*Book* objekat), brisanje knjige iz inventara (po naslovu), pretragu knjiga po naslovu ili autoru i prikaz svih knjiga koje su trenutno dostupne (naslov, autor i godina izdanja).
    - Napisati glavni program koji koristi klasu *Library* za upravljanje inventarom knjiga. Korisnik treba da može da izabere opcije za dodavanje, pregled, uređivanje i brisanje knjiga iz biblioteke. Prilikom dodavanja knjige, program treba da omogući unos svih potrebnih informacija o knjizi. Prilikom uređivanja knjige, korisnik treba da može da izmeni bilo koji atribut knjige (naslov, autor, godina izdanja, broj kopija). Prilikom brisanja knjige, program treba da ukloni knjigu iz inventara na osnovu naslova knjige. Prilikom pretrage po naslovu ili autoru, program treba da prikaže sve knjige koje zadovoljavaju kriterijum pretrage.

8. Potrebno je kreirati klasu **Player** koja ima 5 atributa: **x** (x koordinata, integer), y (y koordinata, integer), width (širina, integer), height (visina, integer), health (životni bodovi, integer)
    a. Potrebno je kreirati konstruktor kojim se definišu vrijednosti svih atributa na zadate vrijednosti. Pretpostavlja se da korisnik unosi ispravno informacije (ne treba raditi validaciju). Svi atributi su **privatni.**
    b. Potrebno je kreirati odgovarajuće **getere** i **setere** za sve atribute Pri postavljanju vrijednosti atributa health onemogućiti postavljanje na vrijednosti koje su manje od 0, a veće od 100.
    c. Potrebno je kreirati klasu **Enemy** koja ima takođe 5 atributa: x, y, width, height i damage (vrijednost napada, integer)
        i. Potrebno je kreirati konstruktor kojim se definišu vrijednosti svih atributa na zadate vrijednosti. Pretpostavlja se da korisnik unosi ispravno informacije (ne treba raditi validaciju). Svi atributi su **privatni.**
        ii. Potrebno je kreirati odgovarajuće **getere** i **setere** za sve atribute Pri postavljanju vrijednosti atributa damage onemogućiti postavljanje na vrijednosti koje su manje od 0, a veće od 100.
    d. Potrebno je napraviti funkciju **check_collision**(player, enemy) koja provjerava da li se objekti player i enemy sudaraju.
    e. Potrebno je kreirati metod **decreate_health**(player, enemy) tako da se trenutna vrijednost atributa health objekta player smanjuje za vrijednost atributa damage objekta enemy ako se desi detekcija kolozije između player i enemy.
    f. Kreirati bar jedan objekat player i dva objekta tipa enemy

9. Napisati klasu *Turnir* koja se sastoji od sljedećih atributa: *naziv_turnira* (string), *lista_igrača* (lista torki oblika ime, broj_bodova), *broj_rundi* (integer). Potrebno je implementirati sljedeće funkcionalnosti:
    a. *Konstruktor* klase Turnir koji postavlja početne vrijednosti atributa naziv_turnira, lista_igrača, broj_igrača i broj_rundi. Atribut lista_igrača treba da bude prazna lista.
    b. *Getteri* i *setteri* za sve atribute klase. Obratiti pažnju na ograničenja atributa broj rundi čija vrijednost je veća od 0, a manja od 10.
    c. Funkciju *dodaj_igrača* sa parametrom ime_igrača koja dodaje novog igrača (torku) u listu igrača turnira tako da je broj_bodova inicijalno 0.
    d. Funkciju *obriši_igrača* sa parametrom ime_igrača kojom se briše igrač na osnovu unesenog imena.
    e. Funkciju *prikazi_najboljeg_igrača* koja prikazuje ime igrača koji je na putu da osvoji turnir. To je igrač sa najvećim brojem bodova.

f. (*) Funkciju *pokreni_rundu* koja simulira odigravanje jedne runde između dva igrača. Funkcija treba da prikaže ishod runde i par igrača u jednoj rudni. Na kraju treba uvećati broj odigranih rundi za 1. Za rudnu se nasumično iz liste igrača biraju 2 igrača. U jednoj rundi igrač koji ima više bodova pobjeđuje sa vjerovatnoćom od 60% u odnosu na protivnika. Napomena: iz modula random koristiti funkciju random koja vraća vrijednosti između 0 i 1 (random.random()), dok random.randint(a, b) nasumično bira broj između a i b.

10. Potrebno je da kreirate klasu **Color** koja ima tri atributa: red, green i blue. Svaka boja ima vrijednosti između 0 i 255 (uključujući)
    - Potrebno je kreirati konstruktor kojim se definišu vrijednosti sva tri atributa na zadate vrijednosti. Pretpostavlja se da korisnik unosi ispravno nformacije (ne treba raditi validaciju). Svi atributi su privatni.
    - Potrebno je kreirati odgovarajuće getere i setere za sva tri atributa klase koji su definisani. Napomena: pri setovanju atributa potrebno je odraditi validaciju (moguće je unijeti cio broj između 0 i 255). Ako je unos nevalidan štampati odgovarajuću poruku
    - Kreirati metod add_red koji sadrži osim parametra self, i parametar change koji označava za koliko treba izmijeniti vrijednost red atributa. Vrijednosti ovog parametra mogu da budu pozitivne ili negativne vrijednosti. Ako je u pitanju pozitivna vrijednost tada se uvećava vrijednost atributa red za zadatu vrijednost, a ako je negativna, onda se smanjuje. Obavezno provjeriti da li je došlo do ispadanja iz segmenta 0 i 255. (npr. ako je trenutno vrijednost za red = 20, a proslijedimo za change -40, to znači da je došlo do ispadanja iz segmenta 0 - 255, jer je 20 - 40 = -20 što je manje od 0).
    - Potrebno je isto odraditi za green i blue atribute, tj. kreirati metode add_green i add_blue, oba metoda imaju osim parametra self i parametar change koji se obavlja izmjena odgovarajućih boja (u add_green ažurira se green atribut, a u add_blue ažurira se blue atribut)
    - Potrebno je implementirati funkciju __lt__ gdje se definiše ponašanje operatora <. Funkcija treba da vrati True ako je color1 < color2. To je tačno ako su sve vrijednosti atributa (red, green i blue) color1 manje od svih vrijednosti atributa od color2. Potrebno je implementirati funkciju __eq__ koja vraća True ili False u zavisnosti od toga da li su dvije boje iste. Dvije boje su iste ako su sve vrijednosti tih atributa iste.
    - Potrebno je implementirati funkciju __str__ koja vraća format stringa u kome se štampaju boju “red”: red_num, “green”: green_num, “blue”: blue_num

11. (Nastavak prethodnog zadatka). Potrebno je da kreirate klasu **AlphaColor** koja je izvedena iz klase Color (nasleđuje sve iz klase Color), a ima još jedan dodatni atribut alpha.
    - Potrebno je kreirati konstruktor koji poziva konstruktor iz osnovne klase (koristiti super() specijalni metod) koji setuje red, green i blue u klasi Color, a nakon poziva super() u konstruktoru setovati vrijednost privatnog atributa alpha.
    - Potrebno je kreirati geter i seter za alpha atribut. Alpha atribut može da ima vrijednosti između 0 i 1 (float vrijednosti)
    - Potrebno je kreirati metod update_color_by_alpha koji osim self parametra sadrži i parametar alpha. Parametrom alpha se mijenja svaka boja tako što se od trenutne vrijednosti boje oduzme trenutna vrijednost boje pomnožena sa parametrom alpha (ako je boja imala vrijednost 200, a alpha je 0.5, nova vrijednost boje je 200 - 200*0.5 = 100). Iskoristiti setere i getere iz klase Color.
    - Potrebno je implementirati funkciju __str__ koja vraća format stringa u kome se štampaju boju “red”: red_num, “green”: green_num, “blue”: blue_num, “alpha”:alpha_num
    - Potrebno je kreirati bar dvije instance klase AlphaColor i testirati sve metode koje ste implementirali

12. Potrebno je da kreirate klasu **Company** koja ima 5 atributa: name (ime kompanije, string), area (oblast djelovanja, string), employees (lista zaposlenih, svaki zaposleni je dictionary oblika {“name”: “some_string”, “surname”:”some_string”, “salary”:”num” }) i balance (trenutni finansijskih balans kompanije, float number), max_num_of_employees (prirodan broj koji predstavlja koliko zaposlenih kompanija može maksimalno da ima).
    - Potrebno je kreirati konstruktor kojim se definišu vrijednosti svih atributa na zadate vrijednosti. Pretpostavlja se da korisnik unosi ispravno informacije (ne treba raditi validaciju). Vrijednost atributa employees je prazna lista []. Svi atributi su privatni.
    - Potrebno je kreirati odgovarajuće getere i setere za sve atribute osim za atribut employees. Pri postavljanju vrijednosti atributa balance i max_num_of_employees onemogućiti postavljanje na vrijednosti koje su manje od 0.
    - Kreirati metod add_employee sa jednim parametrom employee koji dodaje novog zaposlenog u kompaniju. Zaposleni se upisuje u atribut employees kao dictionary gore navedenog oblika (pomoć: samo je potrebno odraditi dodavanje employee argumenta, koji je dictionary, u listu employees). Zaposlenog je moguće dodati u kompaniju jedino ako se njegovim dodavanjem neće prekoračiti vrijednost atributa max_num_of_employees.
    - Kreirati metod remove_employee sa dva parametra employee_name i employee_surname koji uklanja zaposlenog iz kompanije. Brisanje se radi na osnovu kombinacije imena i prezimena zaposlenog. Pretpostaviti da u kompaniji ne postoje dva zaposlena sa istom kombinacijom imena i prezimena.

- Potrebno je implementirati funkciju __str__ koja vraća format stringa u kome se štampaju informacije o kompaniji. “name”: “company_name”, “area”: “company_area”, “balance”: “company_balance”
- Kreirati metod can_pay_employees koji vraća True ili False u zavisnosti od toga da li kompanija ima dovoljno novca na računu (atribut balance) da isplati sve zaposlene.
- Potrebno je implementirati funkciju __gt__ koja vraća True ili False. Za kompaniju A se kaže da je veća od kompanije B ako kompanija A ima više zaposlenih nego kompanija B.
- Potrebno je kreirati bar dvije instance klase Company i testirati sve metode koje ste implementirali.

13. Potrebno je da kreirate klasu **Student** koja ima 4 atributa: name (ime studenta, string), prezime (prezime studenta, string), predmeti (lista predmeta koje je student položio, svaki predmet je dictionary oblika: {“naziv”:“ime_predmeta”, “ocjena”:”ocjena”, “broj_kredita”: ”broj_kredita_koje_nosi_predmet”}) i godina (godina studija, int number).
    - Potrebno je kreirati konstruktor kojim se definišu vrijednosti svih atributa na zadate vrijednosti. Pretpostavlja se da korisnik unosi ispravno informacije (ne treba raditi validaciju). Vrijednost atributa predmeti je prazna lista []. Svi atributi su privatni.
    - Potrebno je kreirati odgovarajuće getere i setere za sve atribute osim za atribut predmeti. Pri postavljanju vrijednosti atributa godina onemogućiti postavljanje na vrijednosti koje su manje od 0, a veće od 8.
    - Kreirati metod insert_subject sa jednim parametrom predmet koji dodaje novi predmet koji je student pologao. Predmet se upisuje u atribut predmeti koji je lista kao dictionary gore navedenog oblika (pomoć: samo je potrebno odraditi dodavanje argumenta predmet, koji je dictionary, u listu predmeti). Ocjene se unose kao A, B, C, D, E i F.
    - Kreirati metod remove_subject koji ima jedan parametar predmet, a koji omogućava brisanje predmeta iz liste predmeti (atribut klase).
    - Kreirati metod compute_average koji računa prosjek studenta. Formula za računanje prosjeka:

      suma(ocjena*broj_kredita)/ukupan_broj_ostvarenih_kredita

      Napomena: Svaku ocjenu potrebno je pretvoriti u odgovarajuću brojčanu vrijednost (A-10, B-9, C-8, D-7, E-6, F-5). Svaku ocjenu koja je F treba isključiti iz prosjeka.
    - Potrebno je implementirati funkciju __str__ koja vraća format stringa u kome se štampaju informacije o studentu. *“ime”: “ime_studenta”, “prezime”: “prezime_studenta”, “prosjek”: “prosjek_studenta”*
    - Potrebno je kreirati bar dvije instance klase Student i testirati sve metode koje ste implementirali.

14. Potrebno je napraviti klasu **Drzava** koja se sastoji od atributa: name (String, ime drzave), population (Integer, broj stanovnika), border (List String, lista imena država sa kojima se granični konkretna država), cities (List Dictionaries, lista rječnika/objekata tako da jedan ključ predstavlja naziv grada, a drugi broj stanovnika). Potrebno je napraviti klasu Federacija koja se izvodi iz klase Drzava i ima dodatni atribut countries (List String, lista imena država koje se nalaze u federaciji), a nasleđuje sve atribute i metode iz klase Drzava. Potrebno je:
    a) Napisati odgovarajuće konstrukture, getere i setere za klase Drzava i Federacija.
    b) Za klasu Drzava potrebno je implementirati metod za dodavanje nove države u listu border.
    c) Za klasu Drzava potrebno je implementirati metode za štampanje naziva gradova sa najvećim i najmanjim brojem stanovnika.
    d) Za klasu Drzava implementirati metod za štampu tako da se prikazuje ime drzave, broj stanovnika i lista gradova koji se nalaze u toj drzavi (odvojeno zarezom, a između atributa sa ;)
    e) Za klasu Federacija potrebno je implementirati metod za kojim se implementira poređenje federacija i to po broju država u okviru federacija (implementirati specijalni funkcije __lt__ i __gt__)
    f) Za klasu Federacija potrebno je implementirati metod za štampanje naziva federacije, broja država u federaciji, naziva država u federaciji (odvojeno zarezom, a između atributa sa ;)
    g) Potrebno je testirati sve metode za bar po dvije instance i obje klase

15. (Zadaci 15, 16, 17 su uvezani) Potrebno je kreirati klasu **Article** koja je opisana sledećim
    - **Atributima** (svi su private):
        1. **title** (samo alfabetski karakteri, unique - ne u setteru nego pri kreiranju novog Article, do 50)
        2. **author** (samo alfabetski karakteri, ime i prezime autora, ne duže od 100 karaktera ukupno)
        3. **description** (opis, do 300 karaktera)
        4. **category** (kategorija, ne više od 20 karaktera)
        5. **views** (broj pregleda, default = 0)
        6. **comments** (default prazna lista, jedan ili više komentara koji se čuvaju kao niz/lista torki, gdje je prvi element title (dužina stringa ne veća od 50 karaktera), drugi element author (dužina stringa ne veća od 50 karakera) i description (sami sadržaj komentara, broj karaktera string ne veći od 120)

- **Metodama**:
    - Konstruktor koji kreira objekat Article
    - **Geteri** i **seteri** za title, author, description, category views
        A. Pri kreiranju settera odraditi validaciju koja je gore navedena. Ako validacija ne prođe uspješno, korisniku štampati adekvatnu grešku. Obratiti posebno pažnju na title, jer ne mogu da postoje dva članka (news) sa istim title.
    - insert_new_comment (title, author, description)
        A. Omogućava dodavanje komentara. Prije dodavanja komentara odraditi adekvatnu validaciju (title unique).
        B. Ako je unos ispravan u listu se dodaje nova torka oblika (title, author, description). Napomena: author nije obavezan, tako da ako ga korisnik ne unese, default vrijednost je “anonim”
    - delete_comment_by_title (title)
        A. Briše komentar u zavisnoti od vrijednosti title parametra
    - delete_comments_by_author (author)
        A. Briše sve komentare u zavisnoti od vrijednosti author parametra
    - get_comment_by_title (title)
        A. Vraća komentar u zavisnosti od title u obliku { title: [author, description] }
    - get_comments_by_author (author)
        A. Vraća sve komentare kao listu torki u zavisnosti od author (torka izgleda kao pri kreiranju)
    - inc_views (num)
        A. Povećava broj pregleda za zadati broj, ako se ne proslijedi num, broj pregleda se uvećava za jedan (default)
    - __gt__ - article 1 je veći od article 2 ako je broj views za article 1 veći od broj views za article 2
    - __str__
        A. Štampa **Article** u formatu: *title: string, author: string, description: string, category: string, views: int, number_of_comments: int*

16. Kreirati klase **TechArticle** koja proširuje osnovnu klasu Article

- Dodatni atributi (private) u odnosu na osnovu klasu (koristiti super)
    - creation_date (string oblika d/m/y, pretpostaviti da je unos ispravan, tačno 10 karaktera, ako je d ili m manje od 10 dodaje se 0 kao prefiks, npr. 07/06/2019)
    - lang (string od tačno dva karaktera, dozvoljene vrijednosti en i rs)
    - Napomena: category ima default vrijednost “tech”, što znači da pri kreiranju TechArticle korisnik ne unosi kategoriju
- Dodatni metodi
    - Kreirati dodatne getere i setere (za creation date i lang) - voditi računa o validaciji
    - get_comments_by_term (term) - vraća sve komentare čija vrijednost title počinje sa term i to kao listu torki (torka izgleda kao pri kreiranju)
    - __str__ - štampa Article u formatu:
      title: string, author: string, description: string, category: string, views: int, number_of_comments: int, creation_date: string, language: string

17. Korisnik unosi podatke (pomoću input) čime kreira bar 4 Article i 2 TechArticle. Sve ih čuvati u jednoj zajedničkoj listi. Pri kreiranju svake instance potrebno je odraditi validaciju koja je definisana pri definisanju atributa (ne smije se dozvoliti korisniku da unese pogrešne/nevalidne podatke). Ako unese pogrešnu vrijednost atributa (pri svakom unosu atributa, provjeriti ispravnost i odmah, prikazati poruku korisniku da ponovo pokuša da unese neku vrijednost koja je ispravna). **Napomena**: navesti koju kategoriju article može da ima, i ako korisnik unese nešto van skupa koji definišete unaprijed, tražiti od korisnika da pokuša ponovo da unese kategoriju.

- Pri kreiranju Article/TechArticle se ne radi definisanje comments već se to radi naknadno i to tako što od korisnika tražite da odabere članak po naslovu (koristiti getter) i onda za taj Article korisnik dodaje komentar (poziva se metod za dodavanje komentara)
- Potrebno je odraditi filteriranje po kategoriji koristeći **filter** funkciju. Npr. kada korisnik unese “politics” vratiti sve articles čija je kategorija politics
- Nakon toga kreirati dvije liste čiji su nazivi redom:
  i. kategorije_sorted_by_views
  ii. kategorije_sorted_by_comments
  U listi **kategorije_sorted_by_views** upisati sve articles sa zadatom kategorijom (pod c) sortirane po broju pregleda koje imaju (od max do min). U listi **kategorije_sorted_by_comments** upisati sve articles sa zadatom kategorijom (pod c) sortirane po broju komentara koje imaju (od max do min)