Fonis Datageeks

Intro to Python -- Domaći zadatak

1. Izračunati **dužinu stringa**:
   1.1. korišćenjem ugrađene funkcije za to
   1.2. korišćenjem petlje.
   String čija se dužina računa može se uneti sa tastature pri izvršavanju koda.
2. Na osnovu date liste brojeva, kreirati **novu listu** koja sadrži sve brojeve iz prve liste pomnožene sa 3.
3. Na osnovu datog np niza, kreirati **novi niz** koji sadrži sve brojeve iz prvog niza pomnožene za 3.
4. Dat je proizvoljni np niz dužine n. Kreirati novi niz iste dužine koji sadrži iste brojeve pomnožene prvim n brojevima iz **Fibonačijevog niza**.
5. Kreirati **beskonačnu while petlju**. Kako ste je prekinuli?
6. Kreirati dve prazne liste **forward** i **backward**. Napisati petlju koja od korisnika traži da unosi elemente u listu (stringove) sve dok ne unese bar tri elementa ili prazan string. Svaki string koji unese korisnik dodati u obe liste tako da forward sadrži stringove redosledom kojim ih je korisnik unosio, a backward obrnuto, to jest od poslednjeg do prvog unetog.
   Van petlje okrenuti backward listu i i proveriti da li su liste jednake.
   [Ovaj zadatak zahteva korišćenje dve proste metode koje nismo kucali na radionicama -- google it!]
7. Kreirati string, **proizvoljnu rečenicu**. Na osnovu tog stringa, napraviti listu stringova, gde je svaki element jedna reč prvobitne rečenice. Iteriranjem kroz listu, saznati koliko slova **a** ima u svakoj reči. To zapisati u listu listi sledećeg oblika:
   Primer:
   s = 'Python je super, ali domaci smara'
   lista = [['Python, 0], ['je', 0], ['super', 0], ['ali', 1], ['domaci', 1], ['smara', 2]]
8. Kreirati **listu naziva kompanija**. Proveriti koji je najduži naziv iz liste i taj naziv zameniti za 'too much'
   8.1. Proveriti koji naziv ima najviše slova m i zameniti sva slova 'm' sa 'mam'
   8.2. Proveriti koji je najkraći naziv iz liste i zameniti ga tako da bude sa svim velikim slovima.
9. **Mock up podataka** je jako čest u našem poslu, naročito pri učenju ili demonstriranju primera. Kada nam nedostaju neki podaci, možemo ih modelovati tako da liče na stvarne (to jest na ono kako očekujemo da izgledaju stvari podaci). To nam omogućava da istog trenutka krenemo da radimo, čak iako nemamo podatke. Tako možemo završiti čitav analitički proces pre nego da imamo bilo kakve stvarne podatke. Kasnije, kada nam stignu pravi podaci, taj proces možemo samo primeniti. Probajmo zato da modelujemo

set podataka 8000 klijenata jedne banke: o njima znamo visinu mesečnih primanja, iznos kredita koji su uzeli, starost klijenta, pol i da li su isplatili kredit.

9.1. **Visinu mesečnih primanja** modelovati tako da podleže *normalnoj raspodeli*. Samostalno odlučiti koje očekivanje i standardnu devijaciju koristiti kako bi što preciznije opisali primanja klijenata. Kada napravite niz pogledajte minimum i maximum niza, kao i neke druge statističke mere koje smo radili.
Hint: Verovatnoća da vam se pojavi ekstremni broj (jako mali ili jako veliki) je jako mala ali i dalje postoji. Ako vam se to desi, a sigurni ste u vaše parametre, probajte da generišete novi niz vrednosti i verovatno ekstrema neće biti.

9.2. **Iznos kredita** modelovati tako da podleže *uniformnoj raspodeli*. Sećate se, to znači da je ista verovatnoća da se pojavi svaki mogući iznos kredita. Iznos kredita može biti od 50000 do 1000000. Funkcija je np.random.uniform, pogledajte *dokumentaciju* da biste znali kako se koristi. (help, ? ili standardna numpy dokumentacija na webu).

9.3. **Godine klijenta** modelovati kao niz brojeva iz *normalne raspodele*. Probajte da se igrate parametrima tako da što bolje opišete godine klijenata koji uzimaju kredit i da budete sigurni da te godine stvarno imaju smisla.
Proveravajte koliko ste uspešni u tome na osnovu statističkih mera, onih koje smo koristili i drugih koji vam padaju na pamet. Za druge možete saznati funkciju ako guglate numpy [naziv_statisticke_mere_eng]
Hint: Deca i ljudi preko 100 godina ne mogu uzimati kredit. :D

9.4. **Pol** modelovati kao niz nula i jedinica, random brojeva, potpuno slučajnih. (*randint,* google it!)

9.4.1. Dodatno: Probajte da napravite još jedan niz podataka o polu koji bi sadržao 'Zenski' i 'Muski' umesto 0 i 1.
Hint: koristite funkciju astype, a zatim logičke uslove za subsetovanje kao sa radionice.

9.5. Niz podataka o tome **da li su klijenti isplatili kredit** ili ne modelovati tako da bude 30% njih koji nisu vratili i 70% njih koji jesu.
Hint 1: U pitanju je *Bernulijeva raspodela*. Ona se vezuje za Bernulija jer je lik bacao novčić i merio koliko puta pada pismo. Ova raspodela upravo govori o tome: Događaj će se desiti sa verovatnoćom p ili neće sa verovatnoćom 1-p.
Na primer, pašće pismo u 45% slučajeva, a u drugih 55% neće. U našem slučaju, 30% klijenta neće vratiti kredit, a ostalih 70% hoće.
X = { 1 with probability p, or 0 with probability 1−p }
Bernuli je samo specijalan slučaj binomne raspodele kada je n=1. Ako te ovo više zanima, odgledaj Khana, od njega možeš najbolje naučiti statistiku i matematiku.
Hint 2: Postoji funkcija *np.random.binomial*, pogledaj dokumentaciju. Već znaš da je n=1. Naš događaj je “vratio je kredit” , treba samo da definišeš p (verovatnoću pojavljivanja tog događaja) i veličinu niza.

10. Proveriti da li iznos kredita zavisi od toga kolika su primanja klijenata, to jest da li su ove dve osobine korelisane (povezane). Prokomentarisati zašto je dobijen takav rezultat.