# CS324 skriptica

1. Šta predstavlja teorija programskih jezika?

* Teorija programskih jezika spada u discipline informatike, zavisi ali i utiče na matematike, softversko inženjerstvo i lingvistiku. Teorija programskih jezika se bavi dizajnom, implementacijom, analizom i klasifikacijom programskih jezika i njihovim pojedinačnim karakteristikama.

1. Objasnite realizaciju paradigmi i koncepata u kontekstu programskih jezika.

* Svaki programski jezik realizuje jednu ili više paradigmi, a svaka paradigma ima jedan koncept, ili češće, sastoji se od skupa koncepata. Programski jezik je način da se apstraktne karakteristike problema odvoje od realizacije rešenja tog problema.
* Koncept programskog jezika je način da se oblikuje apstrakcija tako da odgovara određenom domenu problema.

1. Koje su faze u razvoju programskih jezika?

* Mašinski kod
* Heksadecimalni zapis mašinskog koda
* Asemblerski jezici
* Makroasemblerski jezici
* Viši programski jezici (algoritamski ili proceduralni, strukturni jezici)
* Problemu orijentisani programski jezici
* U koji sloj spadaju asemblerski jezici?
  + Asemblerski jezici spadaju u sloj nižih jezika, pretposlednji sloj - četvrti sloj.
* U koji sloj spada digitalna logika?
* Digitalna logika spada u prvi sloj.
* Gde spadaju jezici kao što je Python?
  + Poslednji odnosno poslednji sloj.

1. Koja je razlika između prevodioca i interpretera? Šta je kompajler?

* Interpreter prevodi kod liniju po liniju i odmah izvršava, dok prevodioc prevodi ceo kod, pa tek onda izvršava. Kompajler prevodi programski jezik u neki drugi jezik.

1. Kakav nivo apstrakcije ima Python?

* U odnosu na Asemblerske jezike? Više apstraktan
* U odnosu na C? Može biti i na višem i na nižem nivou apstrakcije
* U odnosu na Javu? Sličan nivo apstrakcije
* U odnosu na C#? Sličan nivo apstrakcije

1. Koja je glavna podela koncepata programiranja?

* Imperativno i deklarativno programiranje.

1. Šta predstavlja imperativno, a šta deklarativno programiranje?

* Imperativno programiranje je programska paradigma koja opisuje računanje kao izraze koji menjaju stanje programa.
* Deklarativno programiranje je programska paradigma koja opisuje računanje bez opisivanja kontrole toka.

1. Opisati ukratko i dati neki jezik kao primer za

* Proceduralne programske jezike
  + Paradigma programiranja bazirana na zvanju procedura. Procedure sadrže niz instrukcija koje treba izvršiti. Primer: C, BASIC
* Objektno-orjentisane programske jezike
  + Paradigma programiranja bazirana na konceptu objekata koji sadrže atribute i metode. Primer: Java, CPP
* Logičke programske jezike
  + Paradigma programiranja bazirana na formalnog logici. Svaki program napisan u logičkom programskom jeziku ima set instrukcija u logičkoj formi. Primer: ASP
* Funkcionalne programske jezike
  + Paradigma programiranja u kojoj su programi konstruisani uz pomoć sastavljanja i izvršavanja funkcija. Primer: CPP

1. Šta predstavlja skripting jezik? Kako se izvršava program napisan skripting programskim jezikom?

* Skripting jezik (en. scripting language) je programski jezik čiji se kod najčešće izvršava interpretiranjem. Pored toga, skriptni jezici se koriste najčešće za pisanje manjih programa (**skripti**) koji se brzo pišu i služe za obavljanje manjih poslova.

1. Koje su prednosti, a koje su mane skripting jezika?

Treba koristiti skripting jezike kada:

* Se vrši spajanje postojećih programskih komponenti,
* Su česte promene u aplikaciji,
* Je prisutan grafički korisnički interfejs,
* Se funkcije aplikacije često menjaju,
* Je aplikacija proširiva,
* Aplikacija manipuliše stringovima.

Ne treba koristiti skripting jezike kada:

* Postoje kompleksni algoritmi i strukture podataka,
* Se procesiraju velike količine podataka,
* Su funkcije strogo definisane i stalne.

1. Nabrojati osobine skripting programskih jezika. Kako se dele skripting jezici?

* Dele se na **lepljive** jezike (zbog toga što povezuju tj. lepe softverske komponente npr. povezivanje baze podataka sa serverom) i **ugradive** jezike koji se lako mogu ugraditi (embedovati) u drugi program.
* Skripting jezik se interpretira, ili interpretira instrukcije, on se ne kompajlira u izvorni kod,
* O upravljanju memorijom brine sakupljač smeća (en. garbage collector), pa sam programer ne treba da brine o memoriji,
* Skripting jezik uključuje tipove podataka viših nivoa, kao što su liste, asocijativna polja, itd.
* Skript program može pristupiti modulima napisanim u nižem programskom jeziku (na primer C),
* Koriste se obično za administraciju sistema i brzu izradu prototipova (engl. rapid prototyping),
* Dozvoljava grupisanje tipično korišćenih komandi u batch datoteku za procesiranje,
* Dozvoljava kreiranje novih fleksibilnih i konfigurabilnih alata koji „razumeju“ skripte i alate drugih korisnika
* Neformalan način što se tiče tipiziranja promenljivih (nema razlike između tipova kao što su integer, ﬂoat ili string)
* Funkcije mogu vratiti ne-skalarne vrednosti, kao i nizove
* Ponovljeni zadatak može se obaviti mnogo brže
* Skripte izbegavaju manje greške nedoslednosti grešaka u procesiranju

1. Kakvo tipiziranje ima Python programski jezik? Objasniti.

* Jedna od velikih prednosti skripting jezika u odnosu na (standardni) programski jezik je nizak stepen tipiziranja. Snaga slabog stepena tipiziranja leži u tome što se promenljiva ne mora unapred deklarisati odnosno navoditi tip promenljive, smanjujući nastale greške, pa samim tim programer štedi na vremenu potrebnom za pisanje programa.

1. Kakav je programski jezik Python po licenciranju?

* Python je licenciran od strane Python Software Foundation PSF, koji je kompatibilan sa GNU - general public license

1. Kako biste opisali/definisali Python programski jezik?

* Python predstavlja interpretirani, objektno-orijentisani programski jezik visokog nivoa sa dinamičkim sistemom tipiziranja podataka. Najviše se koristi u web programiranju, data science-u, veštačkoj inteligenciji i naučnim primenama.

1. Da li greška u Pythonu izaziva grešku segmentacije?

* Ne, bag ili loš unos neće izazvati grešku segmentacije.

1. Šta predstavlja greška segmentacije?

* Greška segmentacije dešava se kada program pokuša da pristpi memorijskoj lokaciji kojoj ne treba da pristupi ili da pokuša da pristupi lokaciji na način na koji ne treba da pristupi.

1. Kada su objavljene verzije 1, 2, i 3 Python programskog jezika?

* Python 1 verzija je izdata 1994. godine
* Python V2 je objavljen oktobra 2000. godine
* Python V3 je izdat decembra 2008. godine

1. Šta predstavlja razumevanje lista?

* Razumevanje lista je konstrukt u pojedinim programskim jezicima koji služi za pravljenje listi na osnovu postojećih listi.

1. Šta predstavlja sakupljač otpada?

* Sistem sakupljanja otpada jeste oblike automatskog upravljanja memorijom. Sakupljač otpada pokušava da povrati otpad tj. memoriju koju su zauzeli objekti koje program više ne koristi.

1. Da li postoji kompatibilnost unazad između Python 3.x i 2.x?

* Python 3 je prestao sa kompatibilnošću unazad i programi koji su napisani u Python 2 programskom jeziku ne mogu raditi u Python 3.0, ukoliko se prvenstveno ne modifikuju.

1. Navesti ključne izmene između Python 2.x i 3.x.

* Python 2 je imao print kao ključnu reč, dok je u pythonu 3 potrebno ubaciti zagrade, što pretvara print u funkciju.
* U Python 2.x jeziku razdvojeni su str i unicode tipovi podataka, dok su u Python 3.x jeziku oba formata u str tipu.
* Za razliku od Python 2.x, u Python 3.x jeziku treba da se koristi ključna reč as (catch NameError as err) umesto ranije samo except NameError, err.
* U Python 2.x jeziku funkcija input() je parsirala ulaz u različite tipove podataka, dok se za parsiranje u str tipove podataka sada koristila raw\_input() funkcija. U Python 3.x jeziku, input() uvek parsira ulaz kao str tip podataka.
* Deljenje brojeva python 2:

3 / 2 = 1

3 // 2 = 1

3 / 2.0 = 1.5

3 // 2.0 = 1.0

Dok je u pythonu 3:

3 / 2 = 1.5

3 // 2 = 1

3 / 2.0 = 1.5

3 // 2.0 = 1.0

1. Šta je omogućilo Python jeziku da postane programski jezik opšteg tipa?

* Python programski jezik je poznat po svojoj opštoj primeni u svakom domenu razvoja softvera zato što ima mnogo primena: web development, razvoj igara, veštačka inteligencija i mašinsko učenje, data science i vizuelizacija podataka, konzolne aplikacije, Desktop GUI, aplikacije za preuzimanje podataka sa interneta, biznis aplikacije, audio i video aplikacije, CAD aplikacije, embedded aplikacije, aplikacije za procesiranje slika...

1. Šta predstavlja tip podataka?

* Tip podataka predstavlja klasifikaciju ili kategorizaciju podataka. Predstavlja tip vrednosti koji određuje koje operacije se mogu izvršiti nad tim podacima.

1. Šta radi funkcije type() ?

* Type funkcija vraća tip podatka.

1. Kada string str vraća False kada se pozove bool(str)?

* Jedino prazan string “” vraća False, svi ostali vraćaju True.

1. Kako se vrši indeksiranje u Python-u?

* Indeksiranjem stringova se može izvući konkretni karakter iz stringa u određenom redosledu, korišćenjem srednjih zagrada []. Prvi indeks je 0, a poslednji je n-1. Moguće je i negativno indeksiranje, ukoliko stavimo negativan broj, izvlači se karakter stringa od poslednjeg karaktera.

1. Kako sve možemo izdvojiti željeni opseg karaktera u stringu?

* Ukoliko u zagradama stavimo prvi indeks, pa dvotačku, pa krajnji indeks, onda će se izvući karakteri unutar tog opsega. Moguće je i negativno indeksiranje, ali tada izvlačimo od nazad karaktere.

1. Kako se definiše lista, kako tuple, a kako imenik?

* Lista je definisana srednjim zagradama, a elementi se razdvajaju zarezima. lst = [1, 2, 3]. Tuple podaci se definišu tako što su elementi tuplea u malim zagradama, razdvojeni su zarezima. T\_var = (2, 3, 4). Imenici se definišu velikim zagradama {}, zapete razdvajaju ove parove, a svaki par je povezan dvotačkom. i\_dict = {“Ime” : 1234, “Prezime” : 4321}

1. Kako se pristupa elementu liste, tuple-a, i imenika?

* Elementima liste se pristupa indeksiranjem putem srednjih zagrada. Lst[3]
* Moguće je indeksirati više elemenata liste korišćenjem dvotačke.
* Elementima tupla se pristupa isto kao i kod liste
* Elementima imenika se pristupa sledećom sintaksom : dict[key] = value
* Pristupanje vrednosti imenika indeks\_dict[“ime1”]

1. Da li je lista isto što i niz?

* Liste mogu sadržati različite tipove podataka u sebi, dok niz mora imati promenljive koje su istog tipa.

1. Kako kategorišemo programske jezika po tipiziranju? Dati primer.

* Postoji jako i slabo tipiziranje. Druga kategorizacija je prema statičkom i dinamičkom tipiziranju.

1. Kakav je Python programski jezik po tipiziranju?

* Python koristi dinamičko tipiziranje i jako tipiziranje.

1. Koje su naredbe uslovnog grananja i petlji u Python jeziku?

* If, elif, else, for, while

1. Šta predstavlja struktura podataka?

* Strukture podataka mogu u sebi sadržati iste ili različite tipove podataka. Koriste se za smeštanje skupa povezanih podataka.

1. Kako se razlikuju primitive od ne-primitiva?

* Primitive su takve strukture podataka koje uglavnom sadrže jednu vrednost, u zavisnosti od programskog jezika. Programski jezik ne dozvoljana naknadno menanje načina funkcionisanja primitive od strane programera. Dok su neprimitive strukture koje sadrže u sebi više vrednosti.

1. Šta znači da je promenljiva iterabilna?

* Promenljiva je iterabilna kada možemo proći kroz njene elemente pomoću for ili while petlje.

1. Zbog čega Python ne dozvoljava modifikaciju primitiva?

* Ako bismo promenili primitive promenili bismo kako sam program radi i interpreter ga ne bi razumeo (ako broj postane string računanje neće raditi i slično).

1. Kako možemo implementirati tip podataka niz (array) u Pythonu?

* Nizove možemo implementirati uvođenjem iz modula za rad sa nizovima ili ne-primitivne liste. Moramo reći kog tipa su ti elementi.

1. Kako dodajemo element listi na kraj liste, a kako na željenu poziciju?

* Na kraj liste dodajemo funkcijom append(), na željenu poziciju metodom insert(), gde se kao argument navodi pozicija i sama vrednost.

1. Kako izbacujemo željeni element liste?

* Elemente možemo izbaciti iz liste pomoću remove() ili pop() metode.

1. Koje su funkcije za sortiranje liste? Da li se metoda .sort() može iskoristiti za bilo koju iterabilnu promenljivu?

* Koristimo funkcije sort() - za sortiranje po rastućem redosledu; i reverse() - za sortiranje po opadajućem redosledu. Samo liste je moguće sortirati sa funkcijom sort().

1. Šta su stekovi, a šta redovi? Kako se implementiraju u Pythonu?

* I stekovi i redovi se implementiraju u Pythonu korišćenjem listi.
* Stek sadrži objekte koji se ubacuju i izbacuju po sistemu Last In First Out LIFO.
* Red sadrži objekte koji se ubacuju i izbacuju po sistemu First In First Out FIFO.

1. Kako se graf može implementirati u Pythonu?

* Grad se sastoji od čvorora i veza između čvorova. Može se realizovati u Pythonu kao posebna struktura, kao klasa ili preko lista ili imenika.

1. Kako se definiše nova funkcija u Pythonu?

* Koristimo ključnu reč def, nakon čega navodimo ime funkcije, njene parametre, ukoliko postoje, nakon toga dvotačka da bismo označili da deo koda koji sledi pripada funkciji. Funkcija može imati jednu ili više povratnih vrednosti.

1. Šta je opseg, a šta zivotni ciklus promenljive?

* Opseg (en. scope) promenljive jeste deo koda gde se ta promenljiva može prepoznati. Parametri i promenljive definisani unutar funkcije nisu vidljivi izvan funkcije, zbog toga oni imaju lokalni opseg.
* Životni ciklus promenljive jeste vreme za koje ta promenljiva postoji u memoriji. Životni ciklus promenljive unutar funkcije je onoliko koliko je potrebno da se ta funkcija izvrši.

1. Šta su lambda funkcije?

* Lambda funkcija može imati bilo koji broj parametara, ali se mora izvršiti u jednom redu koda. To su jednostavne funkcije, poznate i kao anonimne, jer nemaju ime.

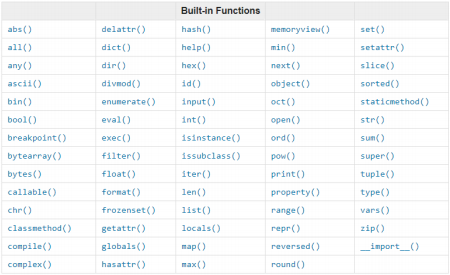
1. Kada se funkciji prenosi parametar, a kada argument?

* Parametar predstavlja promenljivu koja se upisuje unutar zagrada prilikom definicije funkcije.
* Argument predstavlja vrednost koja se šalje funkciji kada se ta funkcija poziva.

1. Da li funkcije u Pythonu mogu vratiti više od jedne vrednosti?

* Da, u Pythonu 3.x, bitno je kako odvojiti te argumente koji se vraćaju, jer se mogu vratiti u više neprimitivnih tipova podataka.

1. Šta predstavljaju ugrađene funkcije?

* Ugrađene funkcije jesu funkcije koje su deo Python interpretera i one su uvek dostupne.
* 

1. Opisati pojmove:

* Enkapsulacija (en. encapsulation)
* Različiti objekti unutar jednog programa u opštem slučaju moći će da komuniciraju jedan sa drugim. Ukoliko programer želi da zaustavi objekte da međusobno interaguju, onda te objekte treba enkapsulirati u odvojene klase. Kroz proces enkapsulacije, klase ne mogu da promene ili interaguju sa specifičnim promenljivama i funkcijama objekata. Slično kapsuli koja zadržava sadržaj unutar ljuske, princip enkapsulacije pravi zaštitnu barijeru oko informacije koja je razdvaja od ostatka koda.
* Apstrakcija (en. abstraction)
* Apstrakcija podataka se može smatrati kao proširenje enkapsulacije zbog toga što sakriva određene osobine i metode ostatku koda u programu, da bi na taj način interfejs ka objektu bio pojednostavljen.
* Apstrakcija podataka je korisna prilikom pisanja programa iz više razloga, ali prvenstveno pomaže izolaciji uticaja promene koda ukoliko se jave problemi unutar objekta, to će uticati samo na promenljive unutar, a ne na kod glavnog programa.
* Nasleđivanje (en. inheritance)
* Koncept nasleđivanja omogućava programerima da prilikom pisanja koda produže funkcionalnost postojećih lasa unutar koda da ne bi došlo do stalnog ponavljanja istih delova koda. Može da se desi da elementi koda uključuju iste osobine, koje poseduju više pojedinačnih metoda. Umesto da se te osobine i metode svaki put ponovo definišu kada se naiđe na element koda koji ih sadrži, mogu se definisati jednom kao generički objekat. Na taj način specifični objekti mogu naslediti osobine i metode, smanjujući ponavljanje koda.
* Glavni objekat postaje superklasa, a izveden objekat postaje podklasa ili izvedena klasa.
* Polimorfizam (en. polymorphism)
* Koncept polimorfizma omoguućava programerima da prilikom pisanja koda pruži mogućnost da jednom napisani kod daje različite rezultate u zavisnosti od ulaza.
* Polimorfizam predstavlja obezbeđivanje jedinstvenog interfejsa prema entitetima različitih tipova.

1. Šta je klasa, a šta je objekat?

* Klasa pruža mogućnost združivanja podataka i funkcionalnosti, predstavlja šablon za pravljenje objekta.
* Objekat je jedna instanca klase, sa definisanim podacima.

1. Šta je imenski prostor u Python jeziku?

* Imenski prostor (namespace) predstavlja mapiranje imena na objekte. Kreiraju se u različitim trenucima i imaju različit vek trajanja.

1. Šta je opseg imenskog prostora? Kako se određuje, a kako koristi?

* Opseg imenskog prostora predstavlja deo koda u Python jeziku u kome je imenski prostor direktno dostupan. Iako se opsezi statički određuju, koriste se dinamički. U bilo kom vremenu tokom izvršenja, postoje tri ili četiri ugnježdena opsega čiji su imenski prostori direktno dostupni:
* • Unutrašnji opseg (en.innermost scope), koji se prvi pretražuje i sadrži lokalna imena,
* • Opseg okružujućih funkcija (en. enclosing functions scope), koji se pretražuje počevši za najbližom okružujućom funkcijom, i sadrži ne-lokalna, ali i ne-globalna imena,
* • Opseg globalnih imena modula koji se pretražuje pretposlednji,
* • Spoljašnji opseg (en. outermost scope) koji sadrži imenski prostori sa ugrađenim imenima i kojie se pretražuje poslednji

1. Kako se definiše konstruktor u Python-u? Kako destruktor?

* Sintaksa konstruktora je :

def \_\_\_init\_\_\_(self):

#telo konstruktora

* Sintaksa destruktora je :

def \_\_del\_\_(self):

print("die")

1. Šta je to prazna klasa? Kada se koristi?

* Prazna klasa je klasa koja ne sadrži nikakve atribute i metode. Nakon toga se naknadno mogu dodati atributi. Međutim, u klasičnom OOP pristupu ovo se ne preporučuje.

1. Šta predstavlja datoteka? Kako se identifikuje?

* Datoteka (en. file) predstavlja skup podataka sadržanu u jednoj celini, identifikovana po imenu i tipu (ekstenziji, en. extension). Datoteka može biti tekstualni dokument, slika, audio ili video podatak, biblioteka podataka, izvršni program ili bilo koji skup podataka. Datoteke se mogu otvoriti, sačuvati, obrisati i premestiti u različite direktorijume unutar sistema datoteka.

1. Šta je klasa, a šta je objekat?

* Klasa predstavlja šablon (en. blueprint) za pravljenje objekata. Klasa pruža mogućnost združivanja podataka i funkcionalnosti. Kreiranjem nove klase kreira se novi tip objekta, koji potom dozvoljava kreaciju novih instanci tog tipa podataka. Svaka instanca klase ima atribute koje su joj dodeljeni. Instance klase takođe imaju metode, koje su definisane unutar klase. Objekat je instanca neke klase.

1. Kako Python deli datoteke?

* Datoteke se najčešće dele na tekstualne datoteke i binarne datoteke. Većina principa važi za oba tipa datoteka.

1. Kako se „ručno“ otvara datoteka, a kako preko kontekstnog menadžera?

* Najjednostavniji način otvaranja datoteke u Python 3.x jeziku jeste korišćenje funkcije open(), koja prima 3 argumenta (od kojih su dva opciona).
* Preko kontekstnog menadžera: Kontekstni menadžer dozvoljava alociranje i oslobađanje resursa u tačno preciziranim trenucima, i automatizuje kontrolu resursa. Upotreba kontekstnog menadžera u Python jeziku postiže se ključnom rečju with. Korišćenje kontekstnih menadžera za rad sa datotekama predstavlja dobru praksu pisanja koda u Python jeziku.

1. Kako odrežujemo delimiter pri radu sa **csv** datotekama?

* Kada kreiramo novi objekat klase csv.writer unosimo dva paramtera, prvi je ime datoteke a drugi je separator(delimiter), koji je najčešće zapeta.

1. Pri obradi izuzetaka, kad se izvršava **try**, a kada **finally** blok?

* U try bloku koda upisuje se deo koda koji može izazvati grešku, a u except bloku vršimo obradu izuzetka. Try se izvršava sve dok se ne pojavi exception, tada se izvršava catch. Kada se završe try i/ili catch, izvršava se finally bilo da postoji greška ili ne.

1. Koja je dobra praksa prilikom hvatanja izuzetaka? Koji izuzeci se prvo hvataju?

* Prvo treba hvatati specifične izuzetke (ValueError kada se ubaci string umesto inta npr), pa tek onda one opšte (Exception, sve ono što nije uspeo da uhvati).

1. Šta predstavlja modul u Python programskom jeziku?

* U Pythonu svaka datoteka sa ekstenzijom .py predstavlja jedan modul.
* Modul predstavlja deo softvera koji poseduje specifičnu funkcionalnost.
* Modul kao i svaka druga Python datoteka može imati promenljive, funkcije, klase sa atributima i metodama koje su već definisane i implementirane.

1. Šta predstavlja paket u Python programskom jeziku?

* Paket predstavlja skup odnosno kolekciju modula. Paket kao takav je zapravo direktorijum Python modula koji takođe sadrži i dodatnu datoteku za inicijalizaciju, da bi se paket razlikovao od običnog direktorijuma koji sadrži Python datoteke.

1. Koja je razlika između modula i paketa?

* Modul je bilo koja datoteka sa ekstenzijom .py, to je deo softvera koji poseduje specifičnu funkcionalnost, dok je paket skup odnosno kolekcija modula.

1. Koja je razlika između paketa i direktorijuma sa .py datotekama?

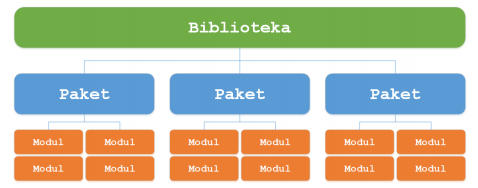
* Paket sadrži i dodatnu datoteku za inicijalizaciju \_\_init\_\_.py, da bi se razlikovao od direktorijuma. Paket sadrži paketni konstruktor.

1. Na koje se načine paket može uvesti u radnu datoteku?

* Na nekoliko načina:
* Import “ime datoteke” npr. Import random i onda unutar datoteke random.randint()...
* Import sa preimenovanjem: import random as rnd
* Konkretno iz modula importovanje: from random import randint (dobro ako importujemo jednu ili dve funkcije ili klase)

1. Šta predstavlja Python biblioteka? Opisati odnos biblioteke, paketa i modula.

* Skup paketa sa modulima i dodatnim podešavanjima predstavlja Python biblioteku. Biblioteka koja se koristi prilikom pokretanja Pythona jeste standardna biblioteka.



1. Šta je Standardna Biblioteka Python programskog jezika?

* Standardna biblioteka se sastoji iz više različitih komponenti. Sadrži najpre tipove podataka koji se mogu smatrati da su srž programskom jezika, npr. brojevi i liste. Takođe sadrži ugrađene funkcije i izuzetke - objekte koji se mogu koristiti u svom kodu pisanom na Python jeziku bez potrebe za naredbom report.

1. Čemu služi PIP?

* Pip menadžer paketa jeste softver koji dolazi uz instalaciju Python programskog jezika sa zadatkom da upravlja eksternim paketima.
* Koristi se u komandnoj liniji i pruža detaljnu kontrolu nad instaliranim paketima.

1. Koje funkcionalnosti dodaju paketi math i datetime?

* Math sadrži matematičke funkcije koje su definisane C standardom, dok datetime uvodi rad sa datumima i vremenom.

1. Šta predstavljaju regularni izrazi?

* Regularni izraz predstavlja niz karaktera koji definišu šablon pretraživanja. Korišćenje regularnih izraza za pretragu teksta ili proveru korektnog unosa teksta predstavlja olakšicu pri programiranju.

1. Za šta se koristi paket NumPy?

* Numpy je jedan od osnovnih paketa za naučno programiranje u Pythonu. Predstavlja biblioteku koja pruža podršku za višedimenzionalne nizove, matrice i grupu rutina za brzu operaciju nad nizovima. Ove rutine su matematičke, logičke, ali i za sortiranje, menjanje oblika, selekciju, kao i za ulaz i izlaz.

1. Šta predstavlja vektorizacija podataka i kako NumPy paket koristi?

* Vektorizacija podataka izdvaja numpy pristup od klasičnog pristupa.
* Vektorizacija jeste mogucnost da se ista operacija vrši paralelno nad većim skupom operanada, odnosno umesto da prolazimo element po element petljom, kažemo niz c = a + b, program će znati da je u pitanju sabiranje po elementu, sabraće svaki element i staviće ga u odgovarajući element novog niza. To radi koristeći već prekompajlovane funkcije u optimizovanom kodu.
* Prednosti:
* koncizniji kod koji je lakši za čitanje,
* manji broj linija koda u opštem slučaju znači manji broj grešaka,
* kod više liči na standardnu matematičku notaciju, što omogućava da se pravilnije kodiraju matematički konstrukti,
* Vektorizacija kao rezultat ima kod koji sam po sebi deluje "Python-ovski". Ovo znači da kod ne poseduje brojne neefikasne for petlje koje se teško čitaju i tumače

1. Da li su NumPy nizovi homogeni ili heterogeni?

* NumPy nizovi su homogeni.

1. Koji je osnovni objekat u paketu NumPy, i koje rutine uključuje?

* Najosnovniji deo NumPy paketa jeste klasa ndarray i njen objekat ndarray. Uključuje sve što bismo inače imali sa nizom. Dodavanje, promena veličine, sortiranje itd. Kada se promeni veličina numpy niza pravi se novi objekat koji će imati tu novu dužinu dok se stari objekat briše, zbog optimizacije memorijskog prostora.

1. Kako se kreira numpy niz?

* Prvo import numpy as np
* a = np.array([1, ,2 ,3 ,4 ,5])
* b = np.array([[1, 2, 3], [4, 5, 6]])
* c = np.array(['Ovo', 'je', 'test', 'string', 'za', 'rad', 'sa', 'numpy', 'paketom.....'])

1. Za šta se koristi matplotlib?

* Matplotlib predstavlja paket za vizuelizaciju podataka kroz grafikone. Proširuje rad sa numeričkim podacima predstavljajući ih kroz različite 2D i 3D grafikone.

1. Za šta se koristi Pillow?

* Pillow je paket koji dodaje podršku za otvaranje, manipulaciju i čuvanje različitih formata slika u Pythonu.

1. Opisati line plot

* Linijski graf predstavlja podatke koristeći liniju.

1. Opisati pie plot

* Pie plot je grafikon u obliku pite, koji parčićima opisuje udeo nekog elementa u odnosu na ostale.

1. Opisati bar i hbar plotove

* Vertikalni bar grafikon prikazuje podatke koristeći pravougaonike odnosno *bars*, gde jedan bar predstavlja jedan tip podatka, a poređani su uspravno.
* Hbar je isti kao i vertikalni bar, samo što su barovi raspoređeni horizontalno.

1. Šta predstavlja SciPy ekosistem?

* To je ekosistem sa paketima i modulima sa naučno programiranje.
* Obuhvata i SciPy paket i njegove module, ali i numpy, pandas, matplotlib…
* DataScience je vezan za biznis aspekt pa se dosta i koristi ovaj ekosistem

1. Koja je razlika između arange i linspace?

* Kod arange imamo korak (step) gde određujemo korak između svakog elementa, ali ne specificiramo koliko elementa će biti
* Kod linspace eksplicitno govorimo koliko je elemenata (default 50), a biće linearna progresija između elemenata

1. Koja je razlika između linspace, geomspace i logspace?

* Funkcijom linspace() vraća se niz sa elementima u datom intervalu sa određenim brojem elemenata.
* np.linspace(2.0, 3.0, num=5) >>>array([2. , 2.25, 2.5 , 2.75, 3. ])
* Funkcijom geomspace() vraća se niz sa elementima u datom intervalu sa određenim brojem elemenata i veličine razmaka između elemenata u geometrijskoj progresiji.
* Funkcijom logspace() vraća se niz sa elementima u datom intervalu sa određenim brojem elemenata i veličine razmaka između elemenata u logaritamskoj progresiji.

1. Koja je razlika između line plot-a i scatter plot-a?

* Line plot je baš linijski plot dok je scatter plot skup tačaka koje su nezavisne, ne povezuje ih linija (primer mašinsko učenje sa onim plotovima, pravi podaci su scatter plot a ona linija je line plot)
* Oba imaju x i y sou.

1. Šta predstavlja funkcija subplot i čemu služi?

* Subpot služi da na jednoj figuri predstavimo više gafikona, tj. da predstavimo više podataka na jednoj slici
* Plotovi se prikazuju po ćelijama

1. Šta omogućava random.seed funkcija?

* Random brojevi jesu slučajni brojevi, ali kada dodamo seed onda podešavamo da budu pseudo slučajni brojevi, odnosno sa nekom ulaznom sekvencom tj. da dobijemo iste slučajne brojeve.
* Dobro kada želimo da tesitramo funkcionalnost nekog programa tj. da upoređujemo sa nekim referentnim vrednostima

1. U kom delu figure se pravi grafikon komandom:

* fig.add.subplot(2,2,1)
* fig.add.subplot(2,2,2)
* fig.add.subplot(2,2,3)
* fig.add.subplot(2,2,(3,4))

1. Koje su faze jednog Data Science projekta?

* Pribavljanje podataka - data capture
* Održavanje podataka - data maintenance
* Procesiranje podataka - data processing
* Analiza podataka - data analysis
* Prikaz podataka - data communication

1. Opisati korake u fazama

* Pribavljanje podataka treba da odgovori na pitanje “Kako se podaci pribavljaju?” i sastoji se od sledećih koraka :

- Skupljanje podataka

- Unos podataka

- Prijem signala

- Izvlačenje podataka

* Održavanje podataka odgovara na pitanje “Šta se dešava sa pribavljenim podacima?” i sastoji se od ovih koraka:

- Skladištenje podataka

- Čišćenje podataka

- Priprema podataka

- Pre-procesiranje

- Arhitektura podataka

* Procesiranje podataka odgovara na pitanje “Šta treba uraditi sa očišćenim podacima?” i sastoji se od ovih koraka:

- Pretraga podataka

- Klasifikacija podataka

- Modelovanje podataka

- Sumiranje podataka

* Analiza podataka odgovara na pitanje “Kako analizirati podatke?” i sastoji se od ovih koraka:

- Istraživanje/potvrdna analiza

- Prediktivna analiza

- Regresija

- Pretraga teksta

* Prikaz podataka odgovara na pitanje “Kako prikazati rezultate?” i sastoji se od ovih koraka:

- Izveštavanje

- Vizuelizacija podataka

- Biznis inteligencija

- Odlučivanje

1. Šta predstavlja Data Science?

* Oblast koja obuhvata rad sa podacima i to je četvrta paradigma nauke.
* Data Science predstavlja multidisciplinarno polje nauke koje se zasniva na naučnim metodama, procestima, algoritmima i sistemima da izvuče znanje iz strukturiranih i (češće) nestruktuiranih podataka.
* Data Science objedinjuje statistiku, matematiku, računarsku tehniku i informacione tehnologije da bi se bolje razumeli nestruktuirani podaci.

1. Šta predstavlja biznis inteligencija?

* Biznis inteligencija predstavlja proces analiziranja i izveštavanja o istorijskim biznis podacima sa ciljem donošenja strateških biznis odluka.

1. Koja je razlika između Data Science i biznis inteligencije?

* Data Science je okrenuta najviše ka budućim podacima i budućim događajima.
* Biznis inteligencija je okrenuta ka istorijskim podacima i prošlim događajima.

1. Koje su dužnosti pozicije Data Analyst / Data Scientist / Data Engineer?

* Data Analyst se smatra početnom pozicijom iz oblasti Data Science. Glavni zadatak je da pregleda podatke organizacije i da iz tih podataka izvuče odgovore na pitanja iz poslovanja. Zatim, treba odgovore da predstavi ostatku tima unutar organizacije. Potrebne veštine su: znanje programskoh jezika Python ili R; znanje SQL jezika; rad sa podacima (pravila pristup, čišćenje, vizuelizacija); statistika i verovatnoća; prikaz analize podataka publici koja nema iskustvo u programiranju i statistici; Tableau/Power BI; SAS/SPSS; Excel
* Data Scientist predstavlja jaču poziciju od analitičara. Mnoge od aktivnost koje radi analitičar treba da zna i naučnik, ali fokus naučnika jeste pravljenje modela i smernica za budući tok poslovanja, ne samo pregled istorijskih podataka. Potrebne veštine uključuju sve veštine analitičara, ali i : razumevanje algoritama nadgledanog i nenadgledanog mašinskog učenja; znanje statistike i mogućnost procene statističkih modela; napredno znanje programskih jezika R i Python, kao i alata kao što je Apache Spark; napredno znanje SAS/SPSS; napredno znanje NPL; napredno znanje Tableau/Power BI
* Data Engineer je zadužen za održavanje infrastrukture podataka jedne organizacije. Ova pozicija zahteva manje znanja iz statistike, ali više znanja iz softverskog inženjerstva i programiranja. Veštine: napredno programiranje u Python jeziku; napredno programersko znanje u jezicima za rad sa velikim skupovima podataka i za kreaciju kanala podataka (Apache Hadoop, Apache Spark); napredno znanje SQL jezika; Postgres; AWS/Azure

1. Šta je aritmetička srednja vrednost / modus / medijana / varijansa?

* Aritmetička sredina predstavlja sumu svih vrednosti podeljenu sa ukupim brojem vrednosti.
* Modus predstavlja vrednost koja se u uzorku ili grupi podataka pojavljuje najčešće
* Medijana predstavlja broj koji razdvaja gornju polovinu uzorka, populacije ili raspodele verovatnoće od donje polovine.
* Varijansa predstavlja odstupanje od srednje vrednosti.

1. Šta predstavlja veštačka inteligencija?

* Veštačka inteligencija (en. Artificial Intelligence, AI) je nauka koja se koristi za konstruisanje inteligencije upotrebom hardverskih i softverskih rešenja.
* Veštačka inteligencija automatizuje ljudsku inteligenciju na osnovu načina na koji ljudski mozak obrađuje informacije

1. Koje su oblasti veštačke inteligencije?

* *robotika* - za pomeranje objekata u prostoru,
* *teorija algoritama* - za konstruisanje efikasnih algoritama,
* *statistika* - za izvođenje korisnih rezultata, predviđanje budućnosti i  
  analizu prošlosti,
* *psihologija* - za modelovanje načina funkcionisanja ljudskog mozga
* *softversko inženjerstvo* - za kreiranje rešenja koja se mogu održavati i koja mogu izdržati test vremena
* *računarska nauka* - za implementaciju softverskih rešenja u praksi
* *matematika* - za izvršavanje složenih matematičkih operacija
* *upravljanje* sistema - za kreiranje feed-forward i feedback sistema
* *teorija informacija* - za predstavljanje, kodiranje, dekodiranje i  
  kompresovanje informacija
* *teorija grafova* - za modelovanje i optimizovanje različitih tačaka u  
  prostoru i za predstavljanje hijerarhija
* *fizika* - za modelovanje realnog sveta
* *računarska grafika i obrada slika* - za prikaz i obradu slika i filmova

1. Kako glasi definicija mašinskog učenja (po Mitchell-u)?

* Za računarski program kaže se da uči da rešava određeni zadatak T, iz iskustva E, i merom performansi P, ukoliko se njegove performanse rešavanja zadatka T, mereno prema P, poboljšavaju kroz iskustvo E.

1. Kako se dele algoritmi mašinskog učenja? Opisati osnovne karakteristike jednog od algoritma.

* **Nadgledano učenje** (en. Supervised learning)
* Kod nadgledanog učenja, algoritmu se pružaju ulazni i izlazni istorijski podaci iz kojih algoritam uči kako da od ulaznih podataka „proizvede“ izlazne podatke.
* **Nenadgledano učenje** (en. Unsupervised learning),
* Kod nenadgledanog učenja, algoritmu se pružaju samo ulazni podaci primera koje treba obraditi, ne i odgovarajući izlazni podaci, te najčešće ne postoji bilo kakva povratna informacija o tome kako treba da izgledaju izlazni podaci.
* Algoritam ima za cilj otkrivanje šablona i grupisanje tih neoznačenih podataka, kako bi otkrio neku vrstu strukture istih.
* **Pojačano učenje** (en. Reinforcement learning)
* Pojačano učenje je model mašinskog učenja koji se bavi načinom na koji softverski agenti treba da preduzmu akcije u okruženju u kome se nalaze kako bi maksimizovali nagradu.
* Pojačano učenje „vuče korene“ iz psihologije ponašanja, te se često kao primer u toku objašnjavanja pojačanog učenja uzima dresiranje pasa. Naime, kad psu kažemo šta treba da uradi, verovatno nas neće odmah razumeti, ali metodom nagrađivanja i kažnjavanja pas polako uči šta je uradio dobro odnosno loše. Primenom iste metode možemo istrenirati programe da izvršavaju različite zadatke, igraju igrice i slično.

1. Koji su algoritmi iz nadgledanog/nenadgledanog/pojačanog učenja?

* Nadgledano učenje: algoritmi klasifikacije i algoritmi regresije
* Nenadgledano učenje: grupisanje(k-srednje vrednosti)
* Pojačano učenje: Markovljev proces donošenja odluke i Kju-učenje

1. Koje su glavne karakteristike linearne regresije?

* Linearna regresija predstavlja osnovni algoritam nadgledanog učenja.
* Kod linearne regresije, na osnovu već postojećih ulaznih i izlaznih podataka, algoritam treba da predvidi izlaz za nove podatke koje su u opštem slučaju realne vrednosti.
* Kao kod svih algoritama nadgledanog učenja, postoji skup podataka (en. dataset) koji se naziva trening skup (en. training set)

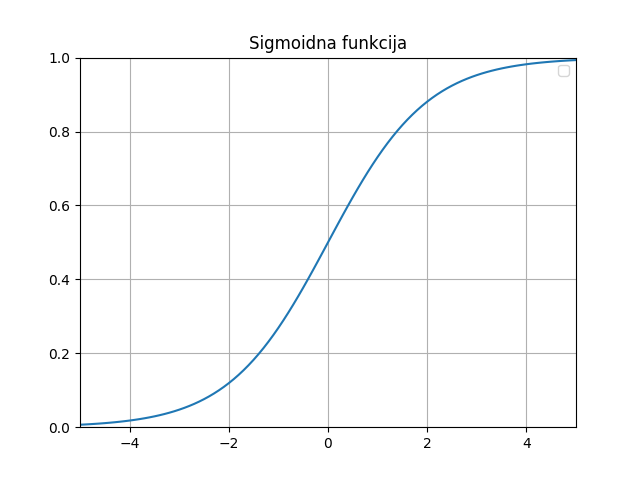
1. Koje su glavne karakteristike logističke regresije?

* Logistička regresija predstavlja osnovni algoritam nadgledanog učenja za diskretne podatke.
* Kod logističke regresije, na osnovu postojećih ulaznih i izlaznih podataka, algoritam treba da predvidi izlaz za nove podatke koje su u opštem slučaju diskretne vrednosti, najčešće 0 i 1.

1. Opisati algoritam opadajućeg gradijenta kod linearne/logističke regresije.

* Najčešće korišćen algoritam za linearne modele mašinskog učenja jeste opadajući gradijent.
* Jedan od najčešće korišćenih algoritama za minimiziranje funkcije greške jeste algoritam opadajućeg gradijenta (en. gradient descent algorithm).
* Koraci algoritma jesu sledeći
* Poći sa nekim početnim vrednostima parametra θ0 i θ1 ,
* Menjati parametre da se smanji funkcija greške J dok se ne dođe do minimuma.

1. Kako izgleda hipozeta kod logističke regresije? Skicirati sigmoidnu funkciju (sa osama)



1. Šta su virtuelna okruženja?

* Virtuelnim okruženjima enkapsuliramo interpreter, odosno pravimo lokalnu kopiju interpretera za projekat koji radimo.

1. Zbog čeka koristimo virtuelna okruženja?

* Koristimo da bismo odvojili od sistemskog interpretera, odnosno kada radimo na nekim većim projektima da razdvojimo dodatne pakete odnosno module

1. Kojom komandom pravimo virtuelno okruženje?

* Iz terminala pythom -m venv naziv

1. Kako znamo da smo u virtuelnom okruženju?

* Znamo tako što stoji ime virtuelnog envrionmenta ispred promta u zagradama (ventv) pa mi kucamo komandu

1. Koje su prednosti mikroservisne arhitekture u odnosu na monolitnu?

* Može se jedan veliki projekat razbiti na mnogo manjih. To omogućava bolje skaliranje projekta. Svaki poseban servis je sopstveni servis koji se može nezavisno razvijati.

1. Koji su nedostaci mikroservisne arhitekture u odnosu na monolitnu?

* Mnogo veci broj interservisne komunikacije što znači da imamo više veza između samih servisa.

1. Šta su funkcije prve klase?

* Funkcija prve klase (en. first class function) jeste sposobnost programskoj jezika da tretira funkciju kao objekat prve klase (en. first class citizen).

1. Šta su objekti prve klase?

* Objekat prve klase predstavlja entitet koji podržava sve operacije koje su dozvoljene ostalim entitetima, i uključuju prosleđivanje kao argument, povratna vrednost funkcije, kao i dodelu vrednosti promenljivoj.

1. Šta je funkcijsko zatvorenje?

* Zatvorenje omogućava funkciji da pristupi uhvaćenim promenljivim kroz kopije vrednosti ili referenci koje se odnose na zatvorenje. Prema wikipediji, pojam zatvorenje, leksičko zatvorenje ili funkcijsko zatvorenje (en. closure), je tehnika za implementaciju leksički obuhvaćenog vezivanja imena u jeziku sa funkcijama prve klase.
* Praktično, zatvorenje je zapis koji čuva funkciju zajedno sa okruženjem. Okruženje je preslikavanje koje povezuje svaku slobodnu promenljivu neke funkcije (promenljive koje se koriste kao lokalne, ali su definisane u obuhvatajućem opsegu), sa vrednošću ili referencom na koju je ime vezano kada je zatvorenje kreirano. Zatvorenje omogućava funkciji da pristupi uhvaćenim promenljivim kroz kopije vrednosti ili referenci koje se odnose na zatvorenje, čak i kada je funkcija pozvana izvan njihovih opsega.

1. Kada se koriste dekoratori i šta su?

* Dekoratorima je moguće proširiti funkcionalnost funkcije, bez modifikacije osnovne strukture funkcije.
* Dekorator (en. decorator) jeste šablon projektovanja u Python programskom jeziku koji omogućava korisniku da doda novu funkcionalnost postojećem objektu a da ne menja njegovu strukturu. Dekoratori se obično javljaju pre definisanja funkcije za koju treba uraditi dekorator.
* Dekoratori se najčešće koriste pri razvoju veb aplikacije korišćenjem Flask paketa ili Django radnog okvira.