Veštačka inteligencija

Projekat „Byte“

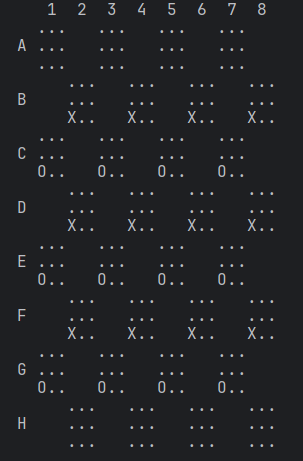
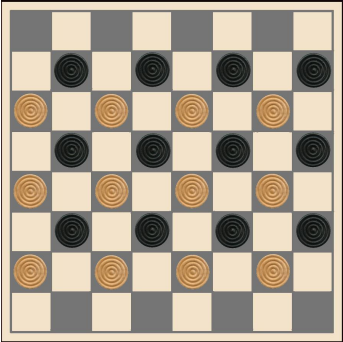
**O igri:**

Byte je strateška igra gomilanja figura unapred postavljenih na tabeli. Tabla je šahovska, kvadratnog oblika sa NxN polja, tako da je N paran broj i broj figura na tabli mora biti deljiva sa 8.

**Pravila igre:**

Dva igrača, crni (X) i beli (O) naizmenično odigravaju po jedan potez. Figure se nalaze na crnim poljima table i kreću se samo dijagonalno za jedno polje. Na početku se figure jednog igrača nalaze u parnim, a drugog u neparnim redovima, pri čemu su prvi i poslednji red prazni. Pobednik je igrač koji složi više „stekova“(*eng. stacks*) od 8 figura, na čijem je vrhu figura njegove boje.

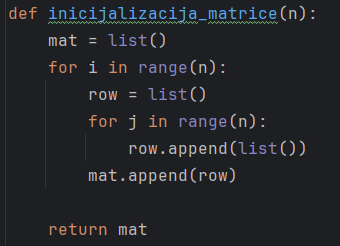
**Početno stanje:**



**Prva faza projekta:**

**Predstavljanje stanja igre:**

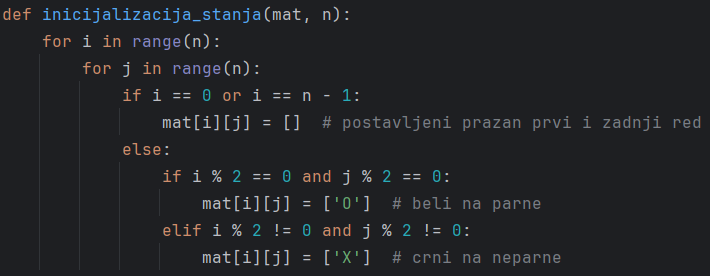
Stanja igre predstavljamo kao matricu(lista liste) čiji su elementi liste koje predstavljaju „stek“, tj. broj figura i njihov redosled slaganja na polju. Funkcija za inicijalizaciju matrice se svodi na jednostavnu ugnježdenu petlju unutar koje se dodaju neophodni elementi.



Funkcija koja generiše matricu 1

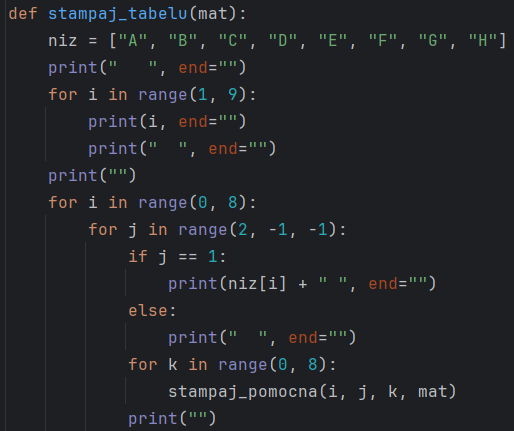
**Postavljanje početnog stanja:**

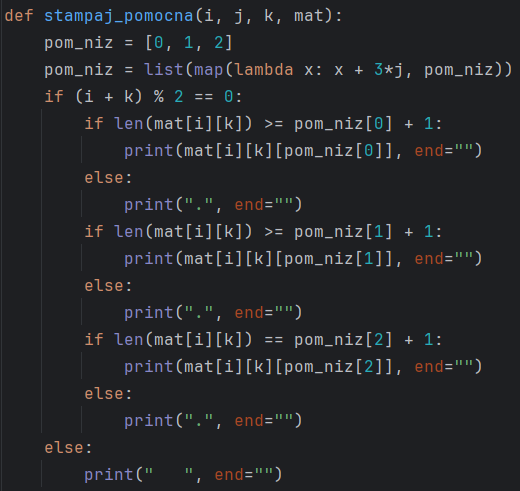
Postavljanje početnog stanja se vrši pomoću funkcije „inicijalizacija\_stanja“, kojoj se prosleđuje inicijalizovana matrica(tabla) i njena dimenzija.



**Prikaz tabele u konzoli:**

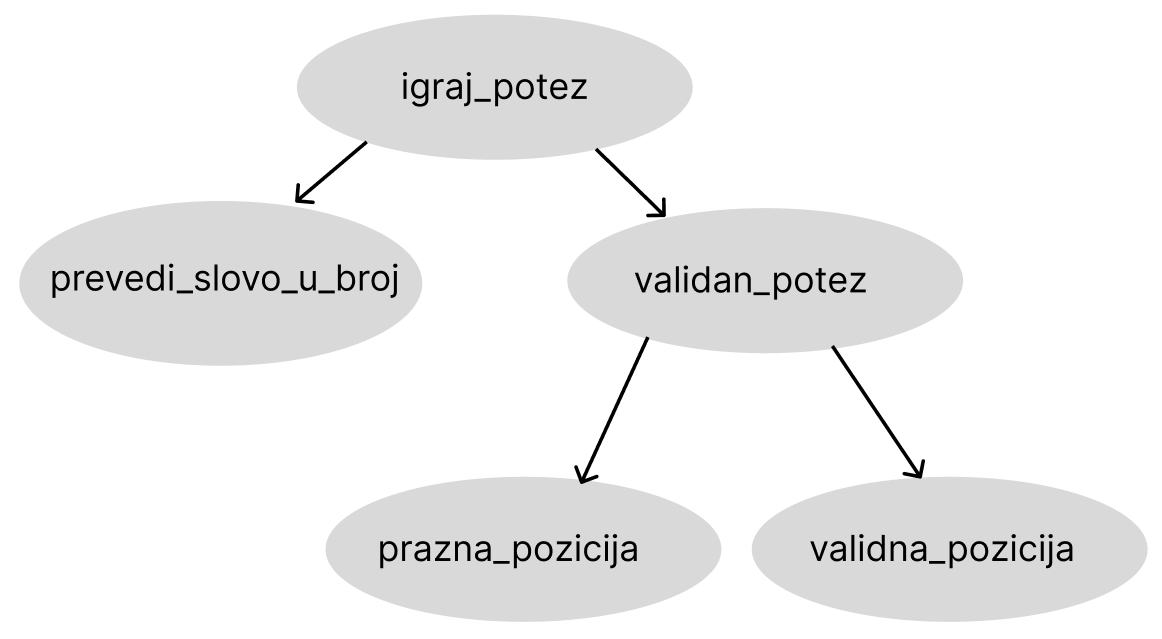
Prikaz tabele u konzoli vrši pomoću funkcije „stampaj\_tabelu“. Funkcija „stampaj\_pomocna“ ima svrhu poboljšanja preglednosti koda. Ona se poziva u okviru funkcije „stampaj\_tabelu“.





**Funkcije za igranje poteza:**

Funkcije za igranje poteza se međusobno pozivaju na sledeći način:



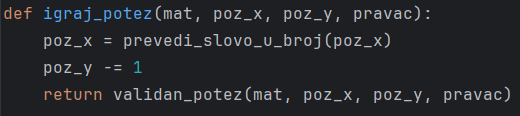
***igraj\_potez***: uzima argumente od igrača i prosleđuje ih dalje funkcijama.

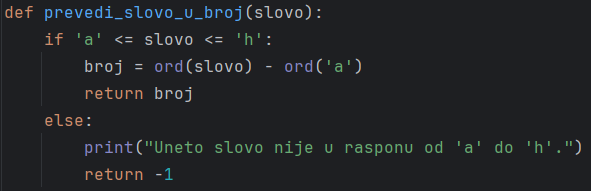
***prevedi\_slovo\_u\_broj***: argument je karakter koji pretvoru u integer vrednost koji predstavlja deo koordinate na tabli (slovo).

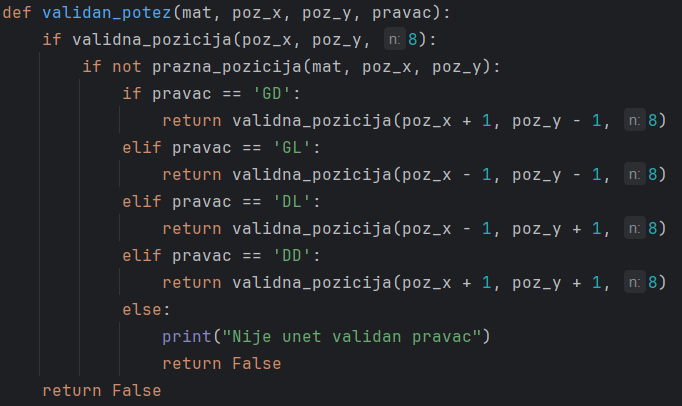
***validan\_potez***: proverava da li je uneti potez u okviru domena pravila igre.

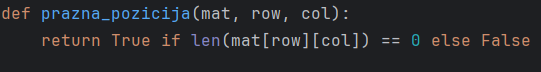
***prazna\_pozicija***: proverava da li je polje na tabli prazno.

***validna\_pozicija***: proverava da li je polje na koje igrač želi da pomeri figuru u okvirtu domena pravila igre.





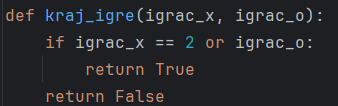






**Funkcija za proveru kraja igre**:

U funkciju za proveru kraja igre se prosleđuju dva parametra: *igrac\_x* i *igrac\_o*. Ova dva parametra predstavljaju broj osvojenih „stekova“ igraca X i igraca O respektivno.

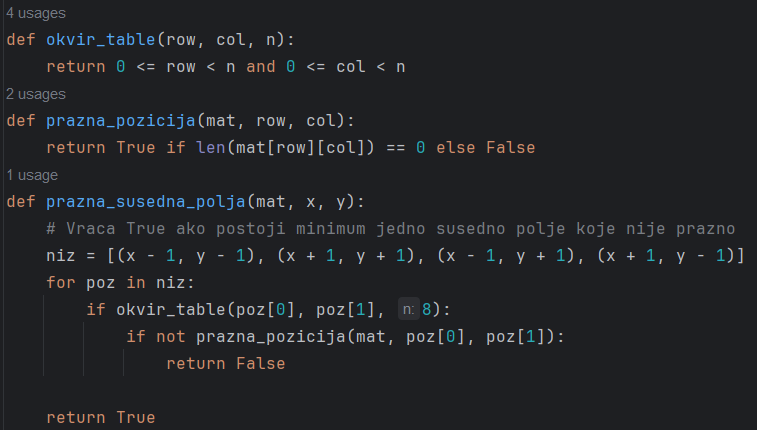


**Druga faza projekta:**

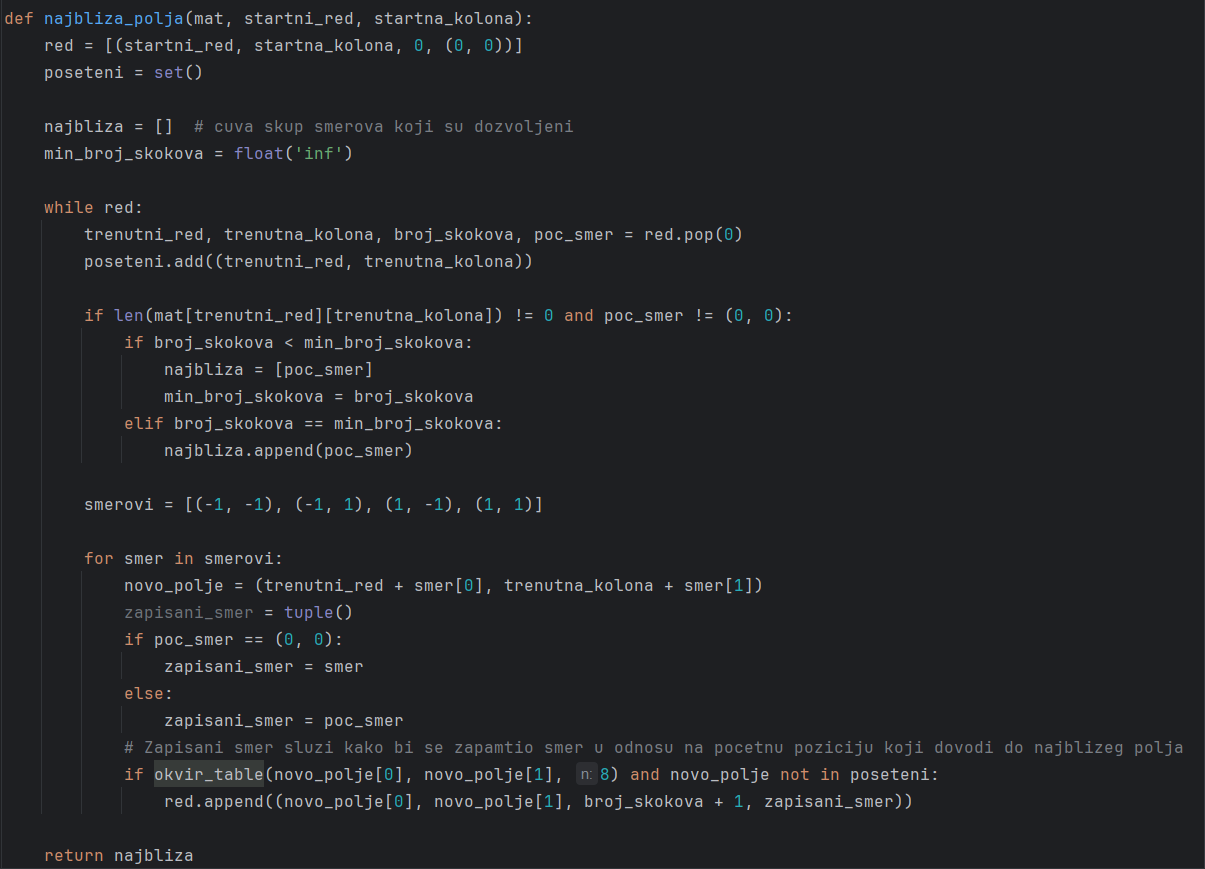
U drugoj fazi se definišu:

1. Funkcije za proveru valjanosti poteza na osnovu konkretnog poteza i trenutnog stanja igre.
2. Funkcije koje na osnovu konkretnog poteza menjaju stanje igre.
3. Funkcije koje obezbeđuju odigravanje partije između dva igrača (čoveka).
4. **Funkcije za proveru valjanosti poteza na osnovu konkretnog poteza i trenutnog stanja igre:**

Funkcije koje proveravaju da li su susedna polja prazna:



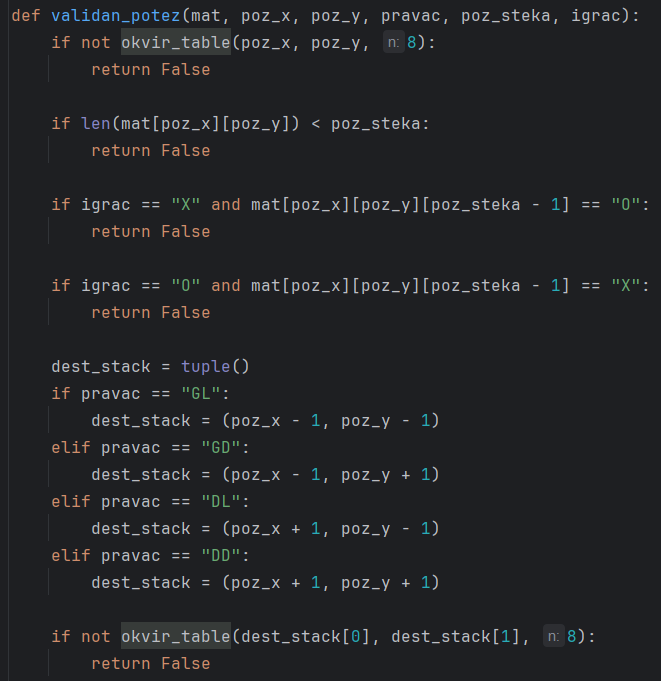
Funkcije koje na osnovu konkretnog poteza i stanja igre proveravaju da li on vodi ka jednom od najbližih figura, odnosno stekova:



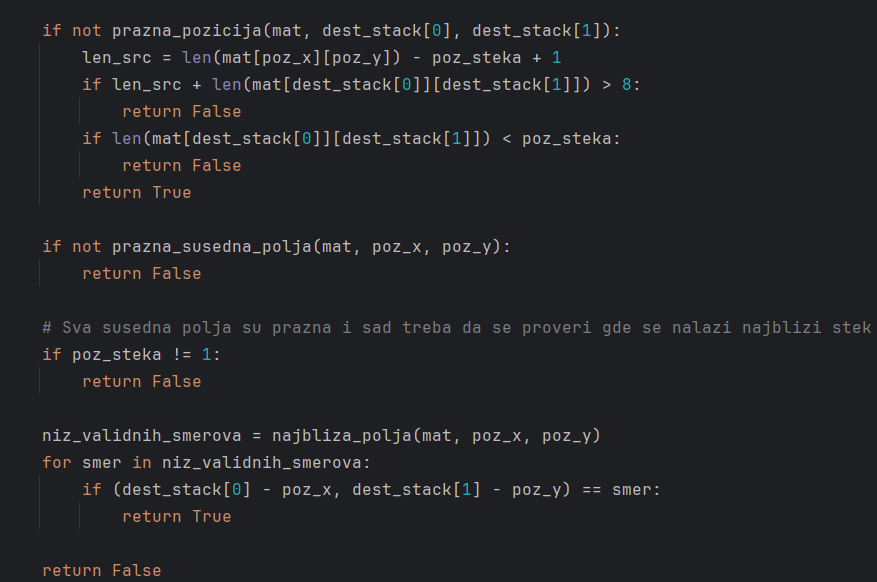
Funkcija *najbliza\_polja()* koristi stablo traženja i metodu BFS kako bi pronašla pravce kojima se stiže do najbližeg nepraznog polja u odnosu na polje koje je prosleđeno unutar funkcije.

Funkcije koje na osnovu konkretnog poteza i stanja igre proveravaju da li se potez može odigrati prema pravilima definisanim za stekove:

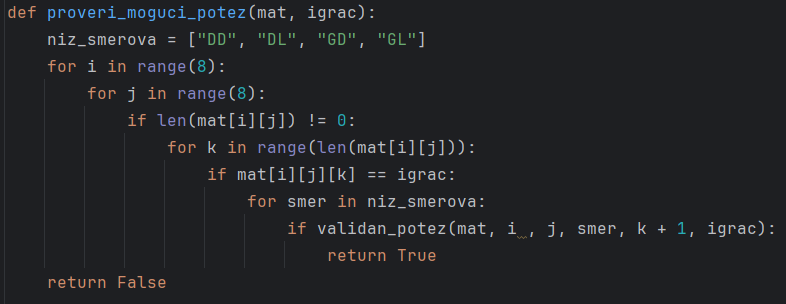
*Prvi deo funkcije validan\_potez*



*Drugi deo funkcije validan\_potez:*



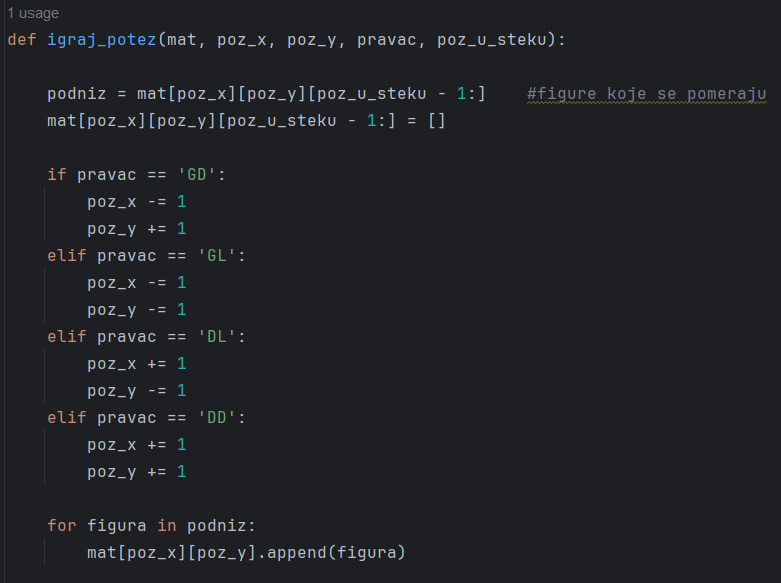
Funkcija koje proverava da li postoji barem jedan potez koji se može odigrati:



1. **Funkcije koje na osnovu konkretnog poteza menjaju stanje igre.**

Funkcija *igraj\_potez* predstavlja operator kojim se menja stanje igre na osnovu konkretnog odigranog poteza. Pozicija sa koje se pomera figura ili stek briše odgovarajući broj figura, a pozicija na koju se prelazi postavlja odgovarajući broj figura na istu.

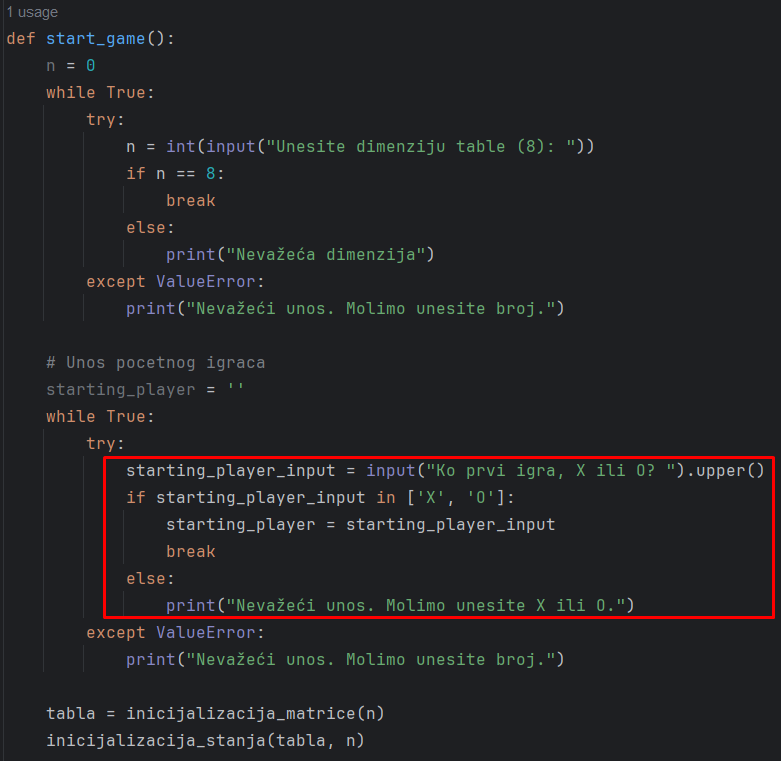
U ovoj funkciji se ne vrši nikakva provera vezana za pravila igre zato što se podrazumeva da je validnost poteza prethodno proverena unutar funkcije *validnost\_poteza()*.



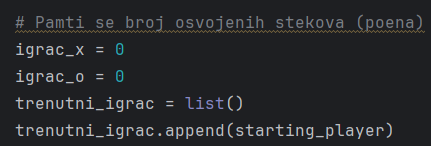
1. **Funkcije koje obezbeđuju odigravanje partije između dva igrača (čoveka).**

Unutar funkcije *start\_game()* se unosi koji igrač igra prvi potez, pamti se trenutni rezultat (broj osvojenih stekova igraca X i O), koji igrač igra trenutni potez i unos poteza (figura i željena pozicija u steku u slučaju da je popunjenost polja veća od 1).

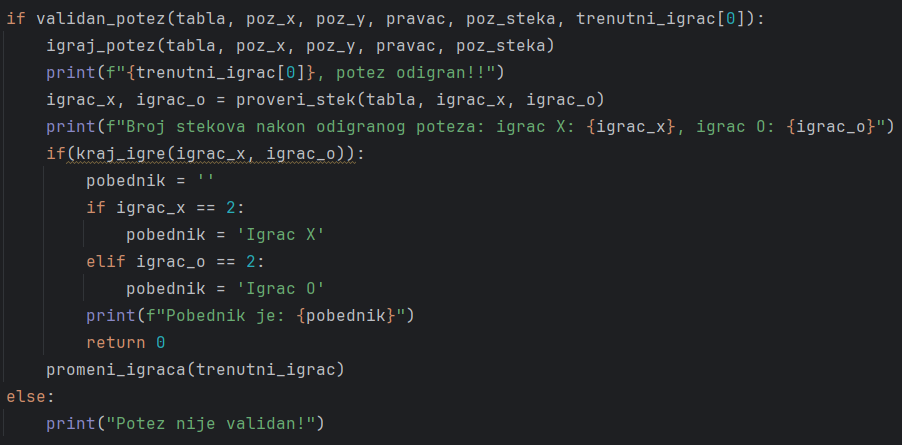
U istoj funkciji se takođe pozivaju funkcije za proveru validnosti poteza i odigravanja istog.



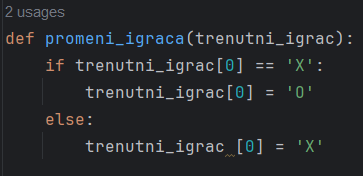
Broj osvojenih stekove (score) i postavljanje trenutnog igrača:



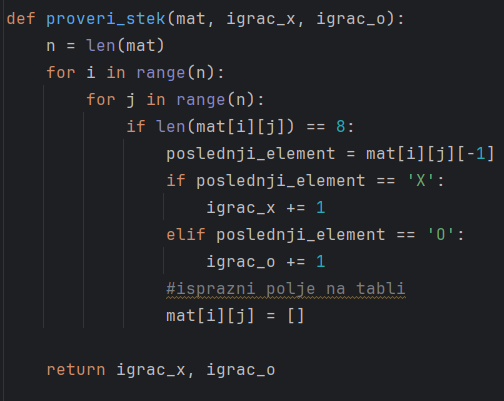
Provera unetog poteza i izvršavanje istog u slučaju da je validan:



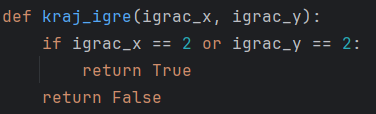
Funkcija za promenu igrača:



Funkcija za proveru dodele poena:



Funkcija za proveru kraja igre:



**NAPOMENA**

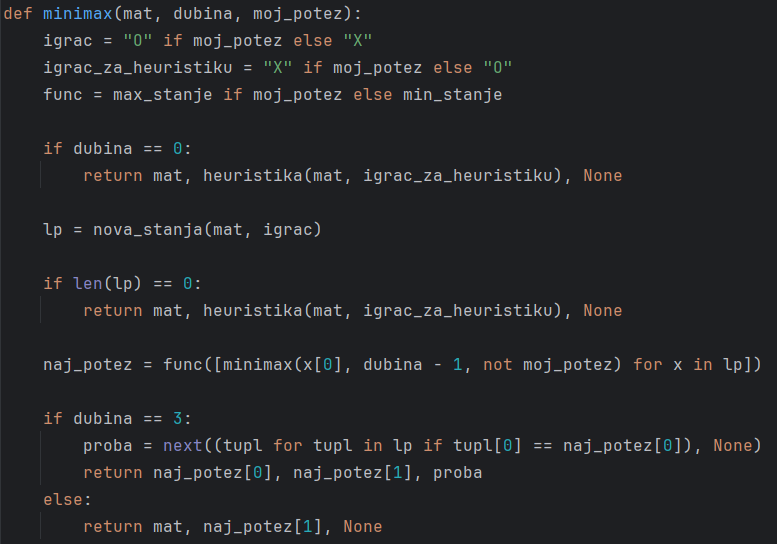
Pri unosu pozicije steka, unose se brojevi od 1 do 7 računajući da broj 1 predstavlja prvu figuru na steku (figuru na dnu steka).

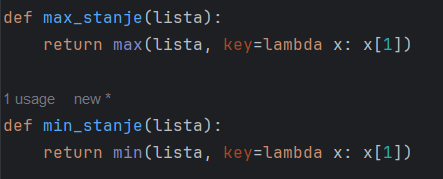
**Treća faza projekta:**

U trećoj fazi su definisane:

1. Implenetacija min-max algoritma za zadati problem.
2. Funkcije koje obezbeđuju odigravanje partije između čoveka i računara.
3. Funkcija koja vrši procenu stanja na osnovu pravila zaključivanja.
4. **Implmentacija min-max algoritma za zadati problem:**

Min-max algoritam se poziva rekurzivno i formira stablo određene dubine. Kada algoritam dođe do korena stabla, koren proverava koje stanje ima maksimalnu heuristiku i vraća informacije o potezu koji dovodi do tog stanja.



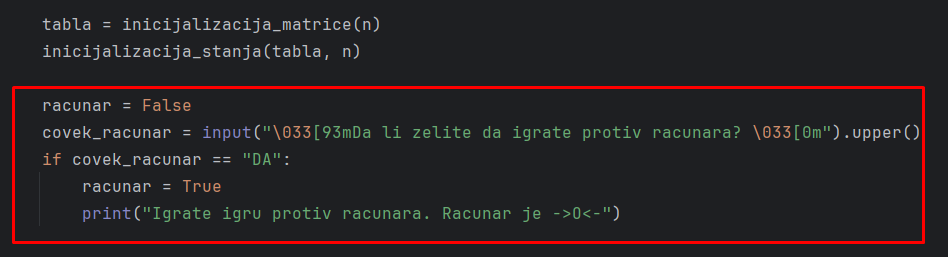


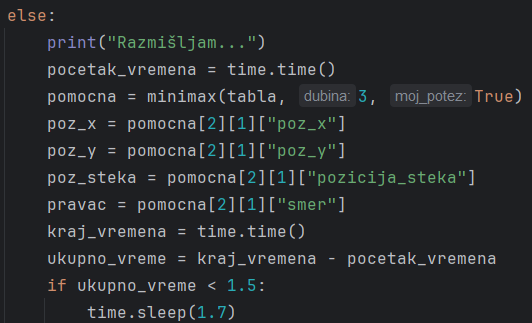
Funkcija *nova\_stanja()*, vraća listu svih novih mogućih stanja u odnosu na stanje koje je prosleđeno i vraća informaciju o tome koji potez je odigran da bi se došlo do tog stanja.



1. **Funkcije koje obezbeđuju odigravanje partije između čoveka i računara:**

Ova funkcionalnost je implementirana u okviru funkcije *start\_game()*. Nakon inicijalizacije početnog stanja, igrač ima opciju da izabere igru protiv računara.



Priloženi deo koda se izvršava kada je omogućena igra protiv računara i kada je računar na potezu.  


Prvo se izvršava „minmax“ algoritam čiji se rezultat upisuje u pomoćnu strukturu, nakon čega se iz pomoćne strukture čita informacija o potezu koji treba da se odigra.

Meri se takođe i ukupno vreme koje je potrebno računaru da izvrši min-max algoritam tako da ukoliko je potrebno manje od 1.5s, onda se izvršava funkcija *sleep()* kako se tabla ne bi prebrzo iscrtala i kako bi korisnik mogao da vidi tablu nakon odigravanja svog poteza.

1. **Funkcija koja vrši procenu stanja na osnovu pravila zaključivanja:**

Logika je implementirana tako da računar bira poteze koji će dovoditi do toga da što više stekova na tabli počinje ili završava njegovom figurom. Takođe računar teži da odigra potez koji će dovesti do formiranja steka od 8 elemenata sa njegovom figurom na vrhu.

