**Projekat iz Veštačke Inteligencije (Byte)**

Preduslovi za pokretanje projekta: Instalacija pygame biblioteke (pip install pygame)

**Ime tima:** CogniStrats  
**Članovi tima:**

Uroš Rakić (18356);

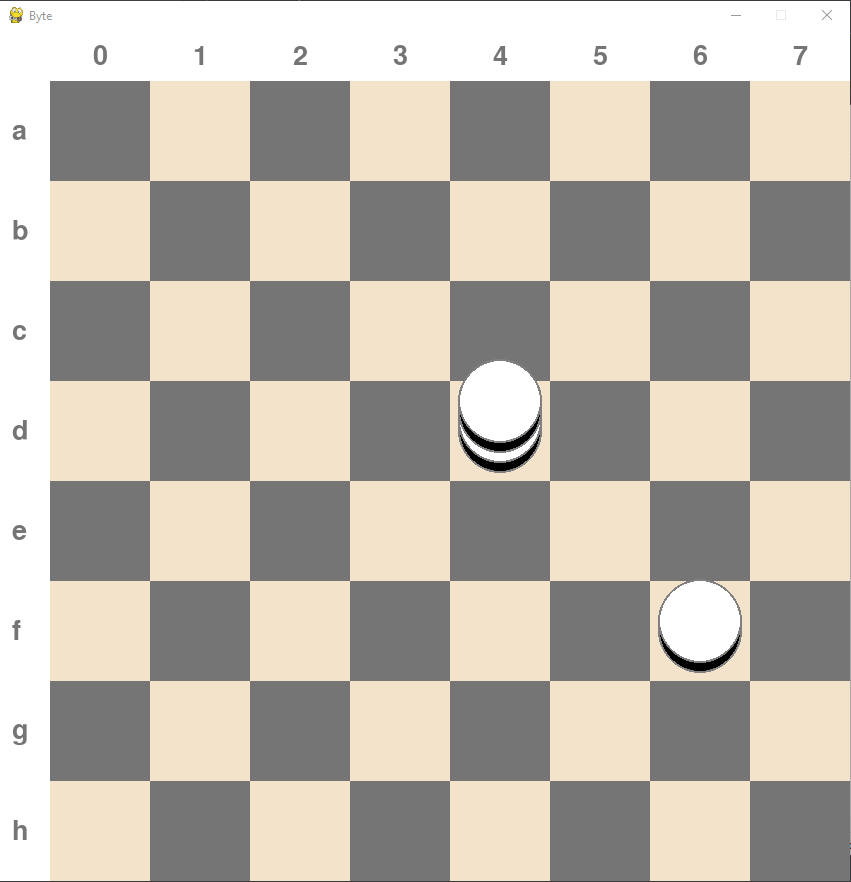
Miloš Jovanović (18196);

Petar Stanojković (18400).  
  
**Formulacija problema**

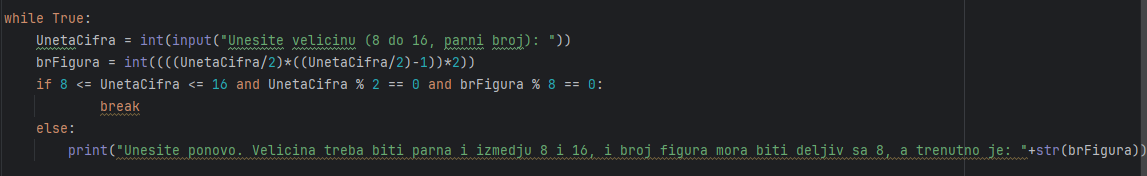
Cilj ovog projekta je razviti digitalnu implementaciju igre "Byte" u programskog jezika. "Byte" je klasična igra na tabli koja uključuje dva igrača koji pomeraju svoje figure (pločice) po šahovskoj tabli sa ciljem kreiranja stack-a od 8 figura gde je vrh stack-a figura sa igračevom bojom.

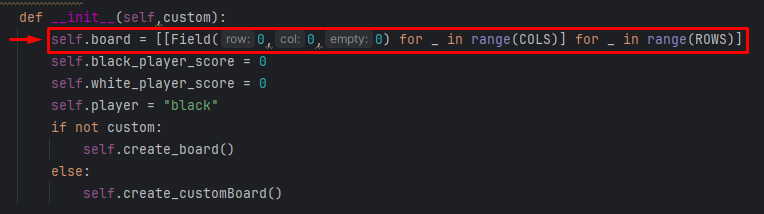
**Problemi i njihova rešavanja**

1. **Definisati način za predstavljanje stanja problema (igre), tačnije: Predstavljanje pozicija figura (stekova) na tabli:**

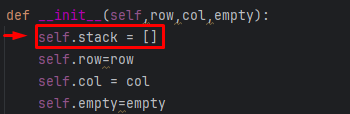


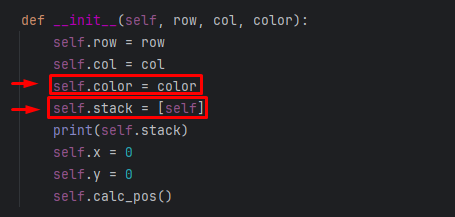
Stanje problema je definisano kreiranjem dvodimenzionalne matrice Filed-ova u klasi Board veličine NxN gde je n broj kolona i redova koje korisnik unosi u constants.





Svako polje(Field) se sastoji iz niza podataka tipa Piece(Figura) što predstavlja stack na tom polju, dok se svaka figura sastoji od svoje boje i svog stack-a(Stack na toj figuri)





1. **Napisati funkciju za postavljanje početnog stanja(Pravljenje stanja igre na osnovu zadatih dimenzija (n) i postavljanje figura na početnim mestima)**

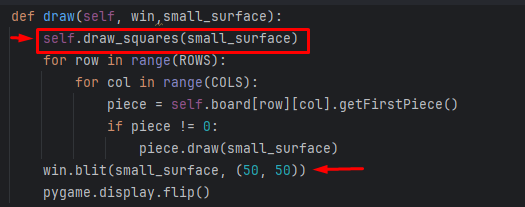
Nakon pokretanja programa korisnik dobija opciju za unos dimenzija polja I ubrzo nakon toga i izbor između inicijalnog stanja i proizvoljnog stanja.



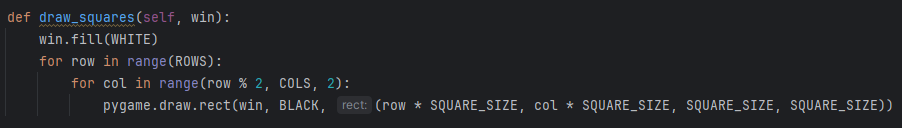


Nakon što korisnik izabere da ne želi proizvoljno stanje sve figure se stavljaju na početna mesta preko funkcija *board.draw()(*Funkcija koja iscrtava tablu na osnovu veličine) i funkcija *create\_board()(*Funkcija koja postavlja figure na početna mesta)

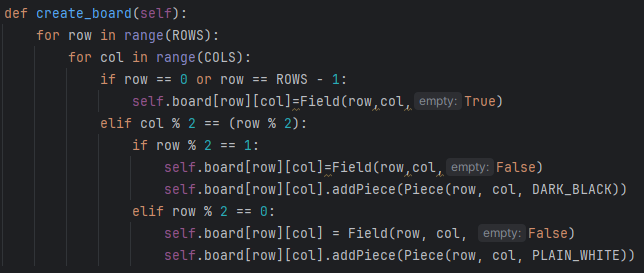
Funkcija *board.draw()*



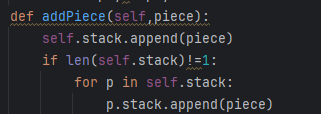
Prima parametar win koji predstavlja ekran I small\_surface koji predstavlja deo gde će se crtati tabla jer se ona postavlja na koordinate 50,50 u ekranu kako bi imalo mesta za oznake redova i kolona. Takođe zove funkciju *draw\_squares()* koja zapravo crta crna i bela polja.



Funkcija create\_board()



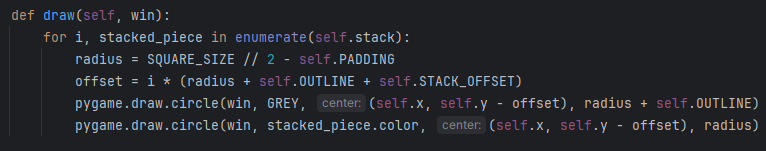
Proverava da li je red prvi ili poslednji ukoliko jeste na to mesto kreira Field(Polje) i dodelju je mu atribut empty na True. Za sva ostala polja kreira Field(Polje) i u njega dodaje Piece(Figuru) uz pomoć funkcije *addPiece()* kojoj na osnovu pravila dodeljuje boju na DARK\_BLACK ili PLAIN\_WHITE



Funkcija addPiece() dodaje u niz figura tu figuru I ukoliko ta figura nije prva prolazi kroz niz svih figura u stacku I svakoj na vrh svog stacka stavlja dodatu, kako bi I svaka figura zanala šta je iznad nje.

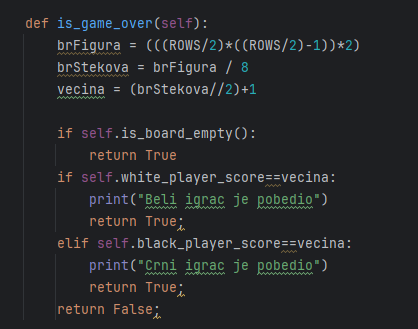
Nakon postavljanja svih inicijalnih figura zove se funkcija board\_draw() koja iscrtava tablu i poziva za svaku figuru funkciju koja iscrtava svaku figuru posebno.(Slika funkcija board.draw() linija piece.draw(small\_surface)

Funkcija draw()(Za piece)

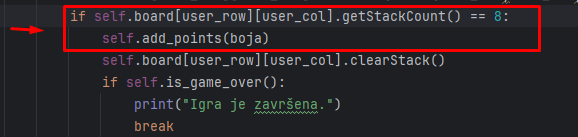


Prolazi stack figure I ukoliko stack nije prazan svaka figura u stacku se crta sa offsetom(Za prvu figuru je offset 0).

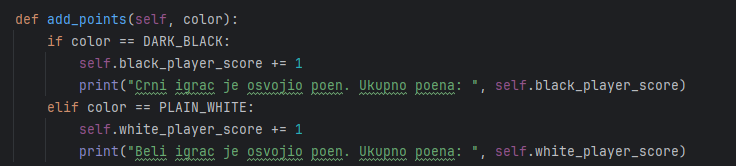
1. **Napisati funkcije za proveru kraja igre, kraj igre nastaje kad je tabla prazna ili igrač u ima u vlasništvu je više od polovine mogućih stekova**



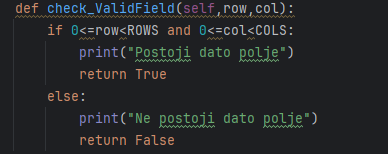
Proverava da li beli ili crni igrač ima većinu stack-ova I proverava da li je tabla prazna I ispisuje odgovarajuću poruku. Poene dodaje na sledeći način



Ukoliko je neki stack došao do 8 dodaje poene igraču čija je figura poslednja na stacku preko funkcije add\_points(boja)

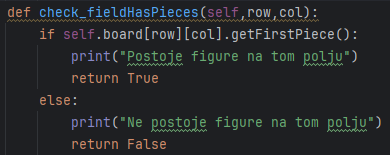


1. **Realizovati funkcije koje proveravaju da li je unos poteza tačan**
   1. **Proveriti da li zadato polje postoji na tabli**



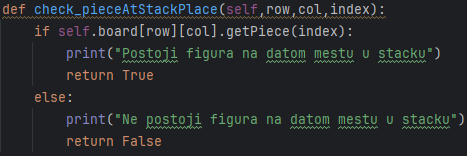
Proverava da li su red i kolona u granicama i vraća odgovarajući Boolean uz štampanje poruke.

* 1. **Proveriti da li postoje figure na zadatom polju**

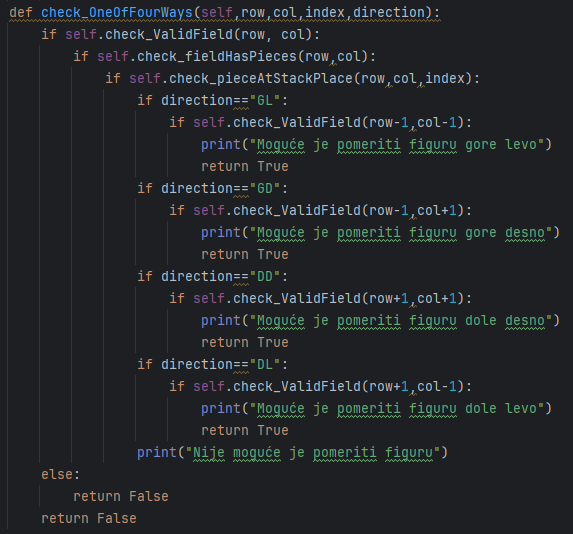


Proverava da li se na tom mestu u redu koloni nalazi neka figura pozivajući funkciju getFirstPiece() koja vraća prvu figuru ukoliko postoji ili 0 ukoliko ne postoji figura. Na kraju vraćamo odgovarajući Boolean i štampamo poruku.

* 1. **Proveriti da li postoji figure na zadatom mestu na steku na zadatom polju**

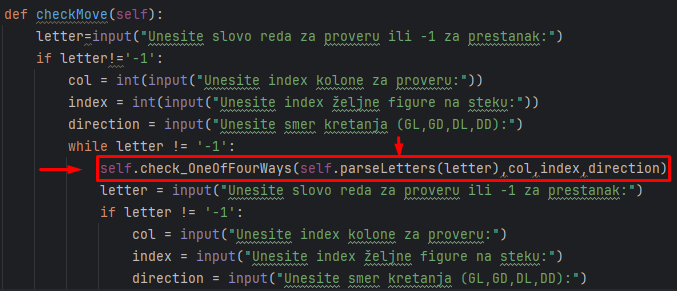
Proverava da li na zadatom mestu u stacku postoji figura pozivanjem funkcije getPiece(index) koja vraća figuru ukoliko postoji ili 0 ukoliko ne postoji. Na kraju vraćamo odgovarajući Boolean i štampamo poruku.

* 1. **Proveriti da li je smer jedan od četiri moguća**

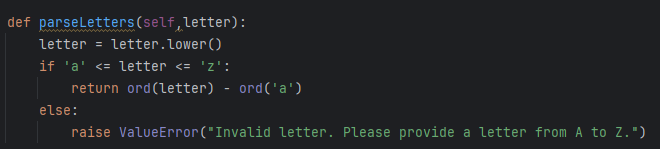


Ova funkcija kombinuje sve predhodne tako što ih redom poziva i ne nastavlja dalje ukoliko vrate False a ukoliko sve prođu onda proverava string koji je user uneo i zove funkciju check\_ValidField(row,col) definisanu u stavci **a.** koja nam proverava da li ne izazimo sa matrice, u odnosu na povratnu vrednost vraćamo Boolean I štampamo odgovarajuću poruku.

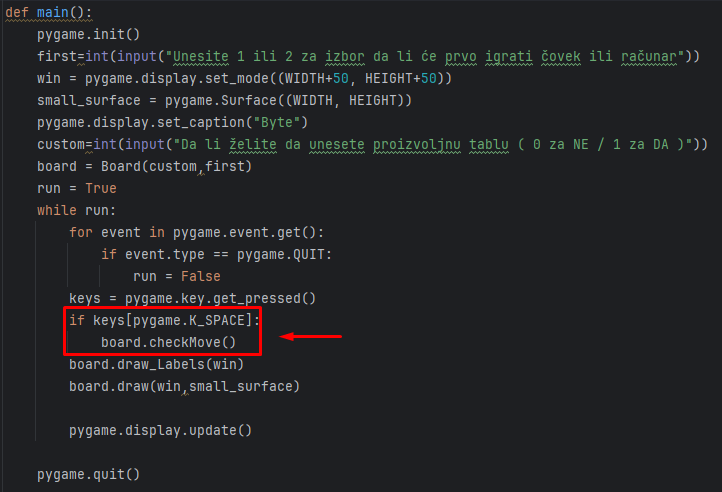
Finalno funkcija checkMove() pravi loop za unos inputa od strane korisinka sve dok on ne unese -1 kako bi proverila potez tako što za svaki korisnikov potez zove funkciju check\_OneOfFourWays()



Funkcija checkMove() u sebi sadrži funkciju parseLetters koja redove koje korisnik unosi kao slova pravi u int indexe

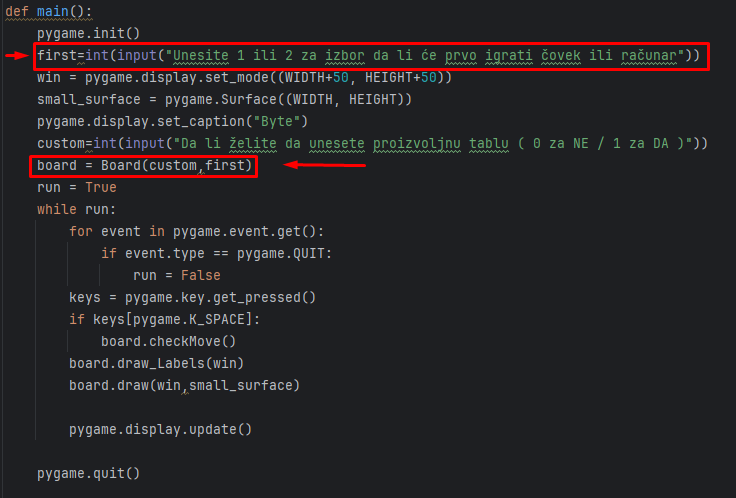


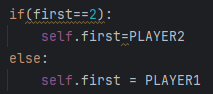
Funkcija checkMove() se poziva u main-u fokusiranjem ekrana gde se nalazi tabla i pritiskom na taster Space tako što pauzira izvršavanje programa sve dok korisnik proverava poteze kako bi omogućio unos u konzoli, unosom -1 se program nastavlja.

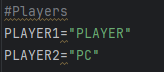


1. **Omogućiti izbor ko će igrati prvi (čovek ili računar)**

Uvek igra prvo crni Igrač može se birati da li će prvo igrati čovek ili računar.





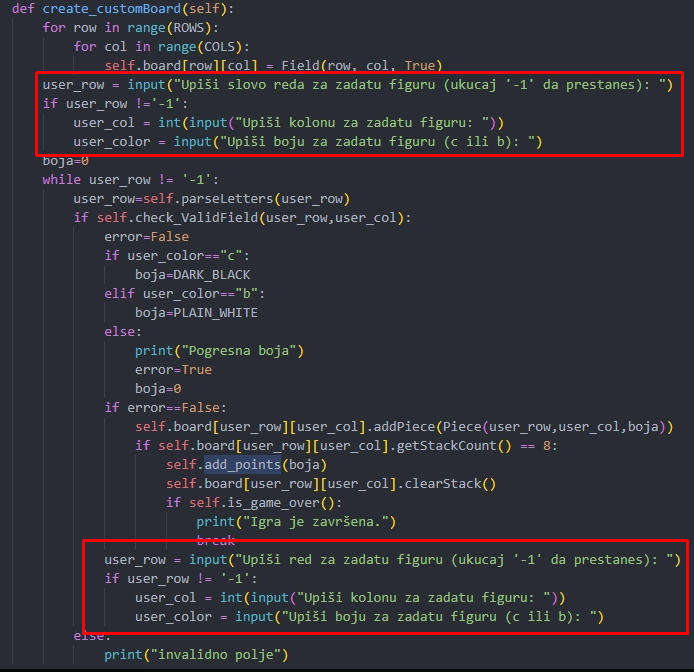


1. **Implementirati funkcije koje obezbeđuju unos početnih parametara igre(Unos dimenzije table (n) i provera ispravnosti unosa)**

Prikazano u problemu **1.**

1. **Implementirati funkcije koje obezbeđuju prikaz proizvoljnog stanja problema (igre)( Prikaz trenutne situacije na tabli sa pozicijama figura)**

Ukoliko korisinik izabere opciju custom tokom kreiranja table (Druga opcija opisana u problemu 2.), umesto funkcije create\_board() pozvaće se funkcija create\_customBoard(), jedina razlika u odnosu na problem 2. Jeste ta što korisnik sam unosi polja i boje figura na tim poljima i onda se samo te figure upisuju u polja(Field) ostatak koda ide istim tokom kao u problemu **2. .** Takođe ova funkcija isto proverava da li je igra završena i dodaje poene igračima pa na taj način je moguće simulirati kraj igre.



1. **Realizovati funkcije za unos poteza pri čemu potez se sastoji od pozicije polja, mesta figure na steku i smer pomeranja (GL, GD, DL, DD)**

Prikazano u problemu **4.**

**Uputstvo za korišćenje**

**1.Kreiranje inicijalne table:**

Pratiti poruke u terminalu I izabrati opciju 0 (NE)

Otvara se novi ekran sa stanjem table

**2.Kreiranje Custom table**

Pratiti poruke u terminalu I izabrati opciju 1 (DA)

Unositi figure odgovaranjem na poruke terminala

Uneti -1 kada je uneto dovoljno figura

Otvara se novi ekran sa stanjem table

**3.Provera poteza**

Ovu proveru je moguće izvršiti nakon otvaranja table ili u koraku 1 ili u koraku 2

Kliknuti na otvoreni prozor sa tablom

Kliknuti dugme “Space“ na tastaturi

Pratiti poruke terminala

Kada se proveri određeni broj figura uneti -1

**4.Kraj igre I poeni**

Moguće je simulirati kraj igre kreiranjem custom tabele tako što će se na jedno polje staviti 2 puta po 8 figura za igrača iste boje.