dsjek za matematiku

Tema: *Igrica Hangman*

*(Vješala)*

**Studenti: Predmetni profesor:**

*Lagumdžija Amina Doc. Dr. Selmanović Elmedin*

*Čengić Adna*

*Katica Afan*

***Sarajevo, 2015.godina***

***Sadržaj:***

1. *Uvod3*
2. *Historijat4*
3. *Kreiranje igrice5*
4. *Testiranje igrice13*
5. *Zaključak16*
6. *Literatura 17*

***1. Uvod***

U dobu kada je život skoro pa i nezamisliv bez računara, mobitela i Interneta te u dobu kada tehnologija iz dana u dan, iz sata u sat sve više napreduje, programiranje predstavlja profitabilan posao. Skoro pa svakodnevno, ne tako mali broj ljudi igra neku od popularnih igara današnjice na svojim *smart* uređajima pitajući se kako je baš ta igrica napravljena. Zanimljivost igrice,te njena velika prisutnost u svijetu moderne tehnologije glavni su razlozi odabira *HANGMAN-a*.

Cilj ovog rada jeste da pokaže jedan od mnogobrojnih načina kreiranja igrice *Hangman (u našem slučaju igrica je kreirana u programskom jeziku Python, ali kreiranje iste je svakako moguće i u drugim programskim jezicima, sa drugačijim linijama koda),* zatim da na što jasniji načini objasni linije koda, te da pokaže zašto ova igrica već desetljećima jako dobro kotira na vrhu liste popularnosti.

Najprije ćemo se upoznati sa kratkim historijatom igrice, kroz koje faze razvoja je prošla od samog nastanka pa sve do danas, te se bazirati na najbitniji dio rada, objašnjenje samog koda igrice.

Nakon čitanja ovog rada, čitalac će moći stvoriti jansiju sliku o tome kako se kreira određena igrica, te uvidjeti osnove programiranja.

***2. Historijat***

***Hangman (VJEŠALA)*** je igrica najmjenjena za dva ili vise igrača. Kada se igra sa dva igrača tada jedan od igrača smišlja riječ, frazu ili pak cijelu rečenicu, dok drugi igrač ili vise njih pokušava da pogodi o čemu se radi nagađajući slova ili brojeve. Upravo ovaj vid igranja ove igre je smišljen za potrebe održavanja *Tradicionalnih igara (međugradski festival)*, 1894.godine, pod nazivom *“Ptice, Zvijeri i Ribe”, "Birds, Beasts and Fishes."(engl.).*

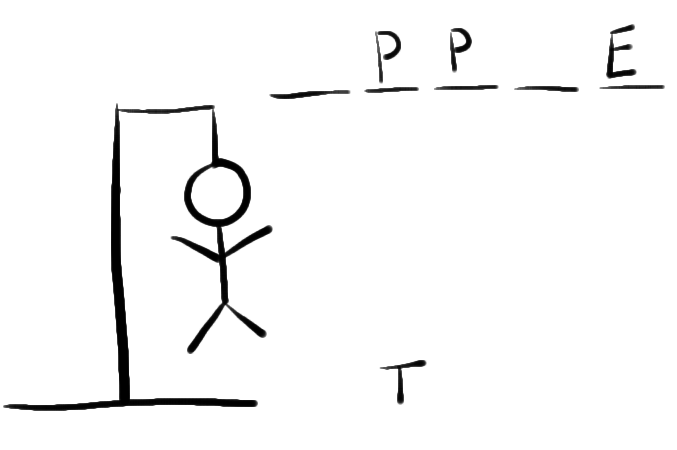
Pravila igre su bila jednostavna; jedan od igrača napiše na dnu papira prvo i zadnje slovo kao pomoć suparniku. Zadatak suparnika jeste da pogodi slova koja se nalaze između. Igrica je mijenjale nazive kroz vrijeme, tako da je od naziva *“Ptice, Zvijeri i Ribe”,* došla donaziva *Vješalo*.

Igrica je 1978., pod pokroviteljstvom [***Speak & Spell***](https://en.wikipedia.org/wiki/Speak_%26_Spell_%28toy%29)sistema za video igrice dobila svoju prvu digitalnu verziju i nosila je naziv *“Čudesni svijest” .*Zanimljivo je da se igrica i dan danas može naći na nekim Internet portalima u svojoj početnoj formi.

Kako je igrica početno bila zamišljena da radi na principu papir-olovka, tj. bila je zamišljena da radi na principu da jedan od igrača zadaje riječ, a drugi pogađa o kojoj riječi se radi, nije bilo teško implementirati ovu ideju u rad nekog od programskih jezika. Upravo je računar bio taj koji je birao riječ, a korisknik u interakciji sa računarom pogađao o kojoj riječi se radi.

Kako je razvoj tehnike sa vremenom rastao, tako je danas došlo do toga da se koristi upravo isti princip rada, uz doradu da neke platforme pružaju korisniku mogućnost da sam bira da li će biti jedan, dva ili pak vise igrača, te da sam izabere težinu igre.

Svjedoci smo sve bržeg razvoja tehnologije, te tako svjedočimo nevjerovatnom razvoju igrice *Hangamana*. Kreatori igrice *(princip papir – olovka)* nisu ni slutili da će njihova igrica kreirana za potrebe festivala dostići današnje razmjere popularnosti.

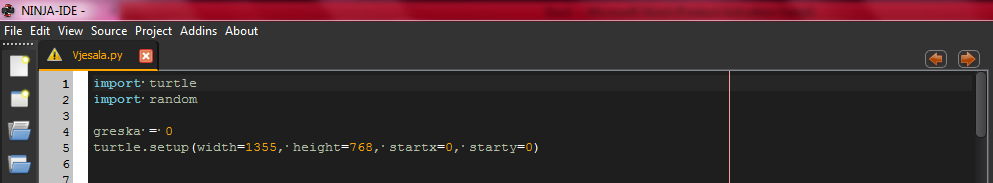


***Slika 1.****Igra prenesena na papir*

***3. Kreiranje igrice***

U nastavku ćemo se upoznati kako je igrica kreirana u promgramskom jeziku Python-u, tj.pojasnit ćemo one linije koda za koje mislimo da su najbitnije.

Prije svega, kako se u ovoj igrici iscrtava vješalo isto je trebalo i napraviti. Zbog toga nam je bila potrebna *Turtle grafika*, koja se odnosi na mogućnost kontrolisanja grafičkog objketa u grafičkom prostoru sa *x + y* koordinatama, pa smo na samom početku uveli *Turtle* te *Random,* koji smo kasnije koristili za nasumični izbor riječi. Također, na samom početku smo uveli i globlanu varijablu, tj. varijablu koja pozivamo onda kada nam je potrebna, pod nazivom *greska* i postavili je na 0. Također smo zbog bolje preglednosti pozor untar kojeg se iscrtava vješalo postavili na prikladne dimenzije.

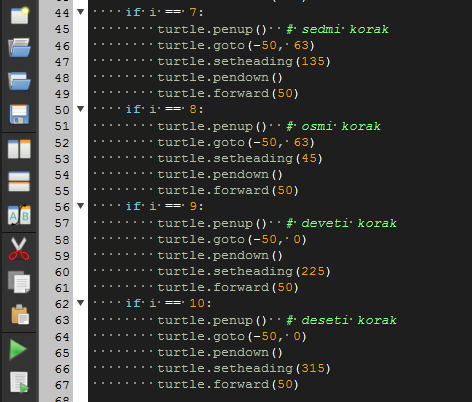
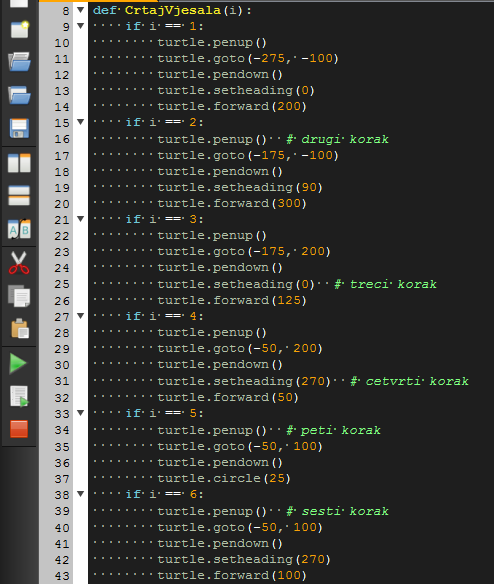


***Slika 2.****Prikaz prvih linija koda*

Nakon što smo uveli ***Turtle grafiku***možemo preći na crtanje i samog vješala. S obzirom na težinu riječi koje smo izabrali dopustili smo da broj grešaka koje igrač može napraviti bude jednaka 10. Zbog toga naše vješalo kreće da crta prvo “postolje” , pa ga kasnije nadopunjuje, tj. crta ga do onog momenta kada broj grešaka bude jednak 10 i nakon toga se obustavlja postupak crtanja. Sa***i*** smo označili funkciju za crtanje dijelova vješala, te u nastavku odredilida ukoliko je npr. i = = 1*turtle* treba da crta postolje, zatim da iscrta gornji dio vješala itd.

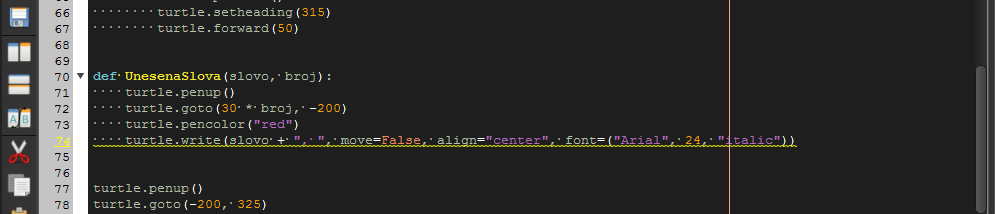
Za iscrtavanje linija koristili smo neke od osnovnih naredbi ***turtle grafike***. Počeli smo od ***turtle.penup()***naredbekoji “naređuje” da strelica prilikom kretanja ne ostavlja trag iza sebe. Zatim ***turtle.goto( -275 , -100)***koji “naređuje” da strelica ide od -275 do -100. Koristili smo i ***turtle.pendown()****.* Upotrebom ove naredbestrelica ostavlja trag iza sebe. Na kraju smo iskoristili ***turtle.setheadinh(0)***, koji služi za postavljanje strelici na 0 stepeni., te ***turtle.forward(200)*** koji crta liniju (idući naprijed) dužine 200.

Sve ove naredbe smo ponovili 10 puta mijenjajući koordinate po potrebi.Također smo povećavali *i* zbog broja dopuštenih greašaka.



***Slika 3.****Korištenje turtle grafike za iscrtavanje vješala*

Nakon što smo formirali vješalo, da bi i naša igrica radila na principu papir – olovka, te radi lakšeg snalaženje unutar igrice odlučili smo da se pri dnu ekrana ispisuju sva unešena slova, bila da su pogođena ili ne. Tu ideju smo također realizovali pomoću*turtle grafike*. Zbog toga smo definisali funkciju *UnesenaSlova(slovo, broj)* i naredili da nam iscrtava slova, crvene boje, formata *Arial*, veličine 24, *Italic* izgleda, koristeći *turtle.write* jer se radi o ispisu.

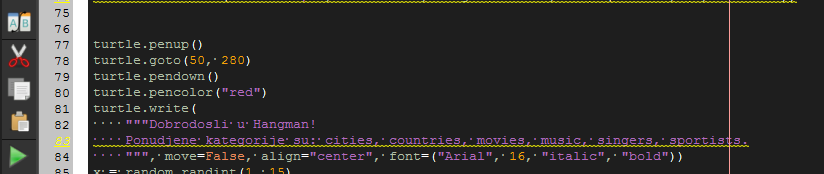


***Slika 4.****Iscrtavanje unesenih slova*

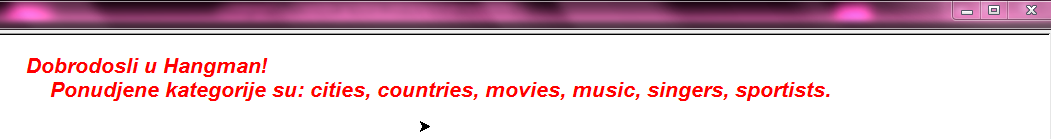
Da bi naša igrica bila što ljepša na izgled, te da bi igrači znali koje kategorije mogu izabarti odlučili smo se da na samom vrhu ekrana bude ispisano:

*“ Dobro došli u Hangman. Ponuđene kategorije su: cities, countries, movies, music, singers, sportists “*

Za potrebe ispisa smo opet iskoristili *turtle grafiku*. Koristili smo već objašnjene naredbe, te pošto se radi o ispisu iskoristili *turtle.write(….)*



***Slika 5.****Izgled linije koda koja se odnosi na ponuđene kategorije*



***Slika 6.****Tekst koji se pojavljuje pri otvaranju igrice*

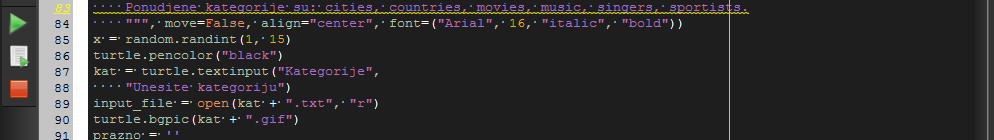
Također kako smo već naveli, da bi igrica bila pokrenuta potrebno je unijeti neku od ponuđenih kategorija, te zaigrati. Bitno je napomenuti da se riječ nasumično biraju pomoću *random-a*. Mi smo se ograničili na po 15 riječi unutar svake kategorije, te smo ih spasili u Notepad-u kao zaseban tekstualni file, tj file koji sadrži znakove u formi linija teksta. Kako tekstualne filove možemo otvarati, čitati, pisati te zatvarati, te kako je nama bilo potrebno otvaranje a zatim i čitanje file-a koristili smo naredbu

***input\_file = open(kat + ".txt", "r")* .**

Također smo željeli, od po mnogima obične igrice napraviti nešto drugačije, pa smo odlučili postaviti slike koje su u bliskom kontaktu sa izabranim kategorijama. Da bi se neka slika postavila u linije koda programskog jezika Python-a potrebno je da budu u *gif* formatu. S toga smo za ubacivanje slike u pozadini koristili naredbu

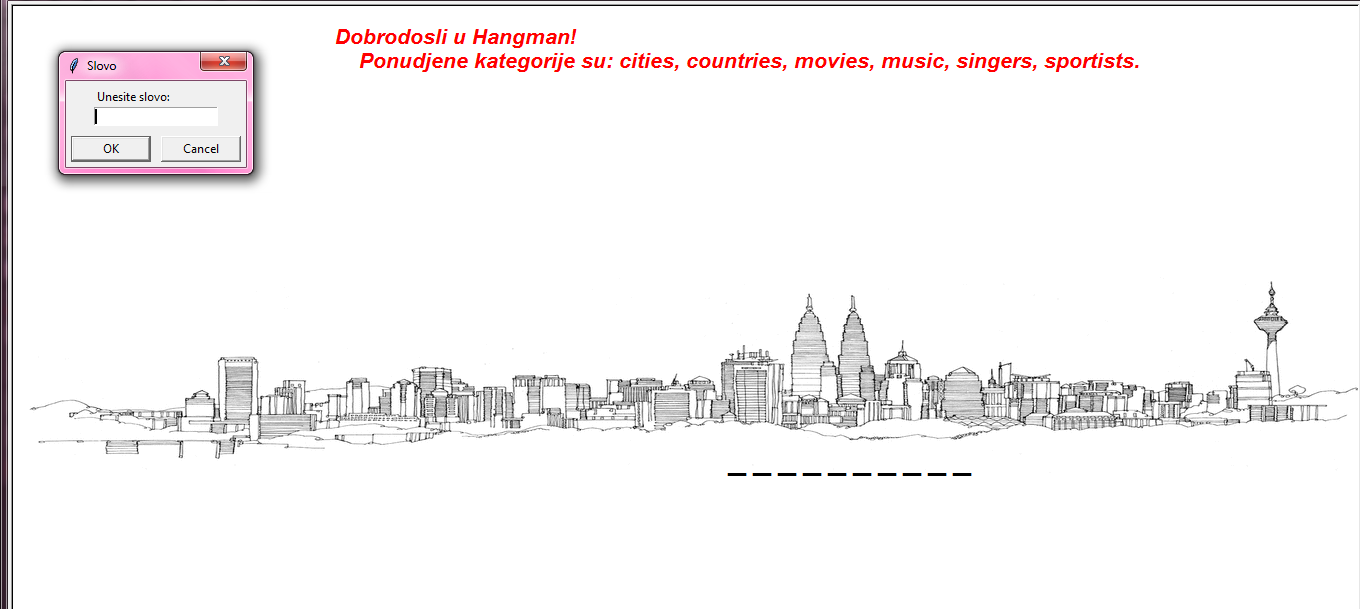
***T***

***urtle.bgpic(kat + ".gif")*.**

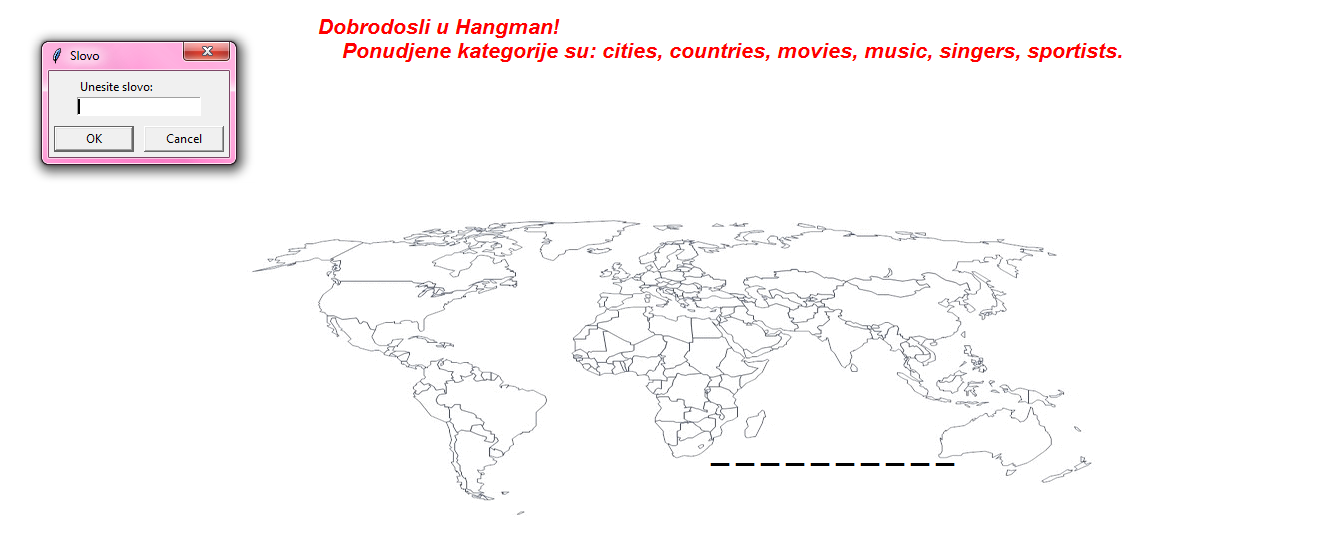


***Slika 7.****Linije koda koje se odnose na dizajn igrice*

Kao što smo već naglasili, da ova igrica ne bi bila obična unjeli smo inovacije koje se nalaze u pozadini.Za različite kategorije koristili smo različite slike.



***Slika 8.****Nakon odabira kategorije* ***cities*** *otvara se prozor sa slike*



***Slika 9.****Nakon odabira kategorije* ***countries****otvara se prozor sa slike*



***Slika 10.****Nakon otvaranja kategorije* ***movies*** *otvara se prozor sa slike*



***Slika 11.****Nakon otvaranja kategorije* ***music*** *otavara se prozor sa slike*

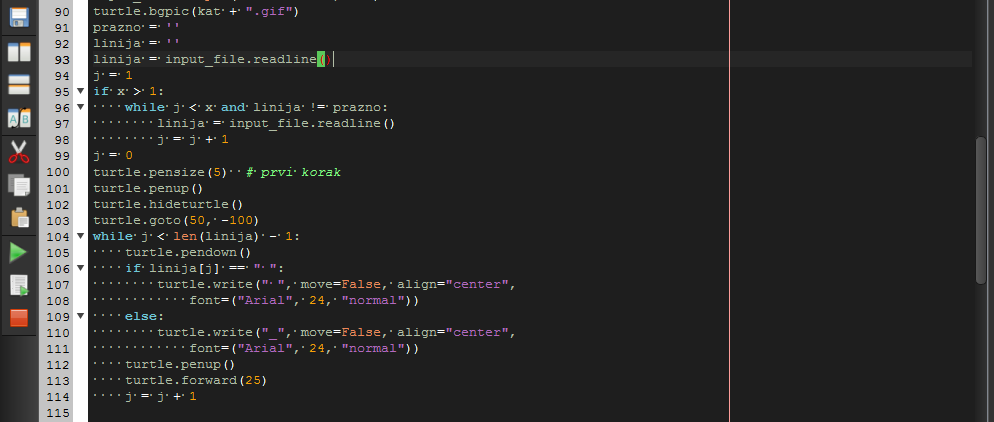


***Slika 12.****Nakon otvaranja kategorije* ***singers*** *otvara se prozor sa slike*



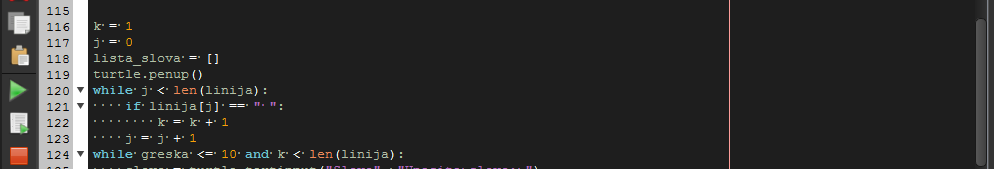
***Slika 13.****Nakon otvaranja kategorije* ***sportists*** *otvara se prozor sa slike*

Nakon što smo postavili pozadinske slike, na red je došlo i crtanje crtica na kojima će biti ispisana slova koja pogodimo, ukoliko ih pogodimo. Kao što smo već par puta naveli riječi se biraju *randomom*.Crtice smo crtali služeći se *turtle grafikom*. Na početku strelicu postavljamo na prvi korak odnosno na sami početak, te koristimo neke od standradnih naredbi *turtle grafike*. Unutar *while petlje* koristimo *if* i*else* naredbe. U *if* uslovu postavljamo da ako su dvije riječi u pitanju između njihide prazno mjesto, te određujemo kojim fontom da riječi budu ispisane. Također koristimo i *turtle.forward* naredbu da bi se crtice pomjerile tj.da ne bi bile jedna na drugoj.



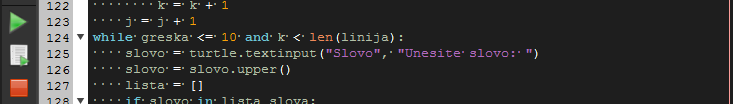
***Slika 14.****Linije koda koje služe za iscrtavanje crtica*

Nakon objašnjenog potupka, kreiramo za početak praznu listu, te*while petlju*. U *while petlji* definišemo da nam se razmaci ili prazna mjesta broje kao slova.



***Slika 15.****Objašnjenje while naredbe*

Koristimo još jednu while petlju u kojoj postavljamo uslov da ukoliko je broj grešaka manji ili jednak od 10, teukoliko je k (dužina riječi) manja od dužine liste da nam se pojavljuje prozorčić u koji unosimo slovo za koje mislimo da se nalazi unutar izabrane riječi.



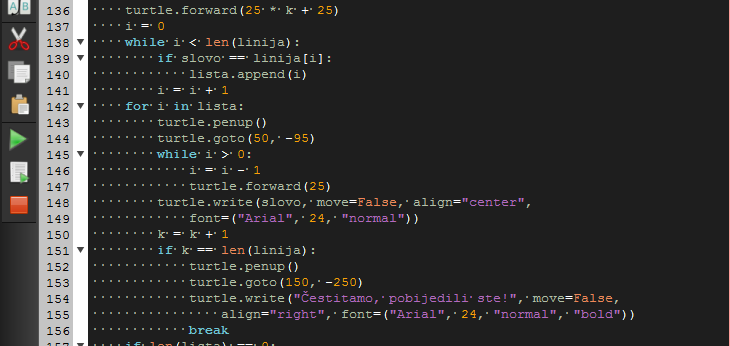
***Slika 16.****Linija koda koja se odnosi na crtanje prozora koji služi za unos slova*

Dalje, korištenjem *if naredbe* želimo da postignemo da ukoliko je neko slovo iz liste već unešeno automatizmom bude preskočeno. Odnosno, da kada unesemo sa tastature neko slovo ono prvo treba da ide u praznu listu, koju smo kreirali i nakon njegovog ponovog unošenja program treba da uoči da je već jednom unešeno i nastavlja dalji rad normalno.



***Slika 16.****Slikovni prikaz gorenjeg pojašnjenja*

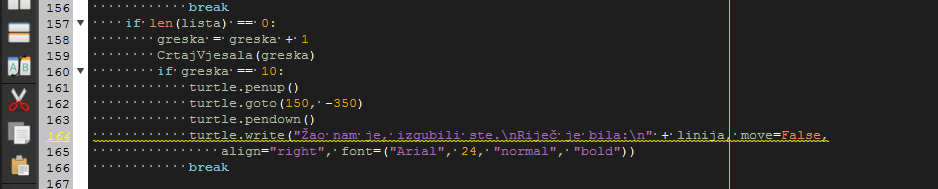
Dalje, pomoću *while naredbom* stvaljamo index od “slovo” u listu. Unutar *for petlje* ponovo koristimo *turtle grafiku* i pomjeramo kornjaču do određenog mjesta na indexu. Također zadajemo da nam kornjača ispisuje slova fonta Arial, veličine 24.Zatim odredimo da ukoliko nam je *k (dužina liste)* jednaka dužini riječi pobjeđujemo i dobivamo ispisanu poruku*“Čestitamo pobjedili ste”.* Upotrebom *break-a* obustvaljamo dalju radnju.



***Slika 17.****Linija koda koja se odnosi na pobjedu u igrici*

Ukoliko je dužina liste jednaka nuli, pozivamo globalnu varijablu *greška* koju smo na početku odredili da je jednaka 0 te je pri pravljenju greške tj unosom pogrešnog slova uvećavamo za 1. Također, izdaje se i naredba da vješalo bude crtano dio po dio.

Kada je broj grešaka dosegnuo 10, ispisuje se poruka *“Žao nam je izgubili ste.Riječ je bila \_\_ “*.Traženu riječ pozivamo iz liste linija.Naravno, i ovdje pozivamo *break* koji prekida dalji rad.



***Slika 18.****Ipis linije koda u slučaju da smo izgubili*

I na kraju, s obzirom da smo i otvorili tekstualni file moramo ga i zatvoriti. Pored toga zatvramo i *turtle grafiku*.

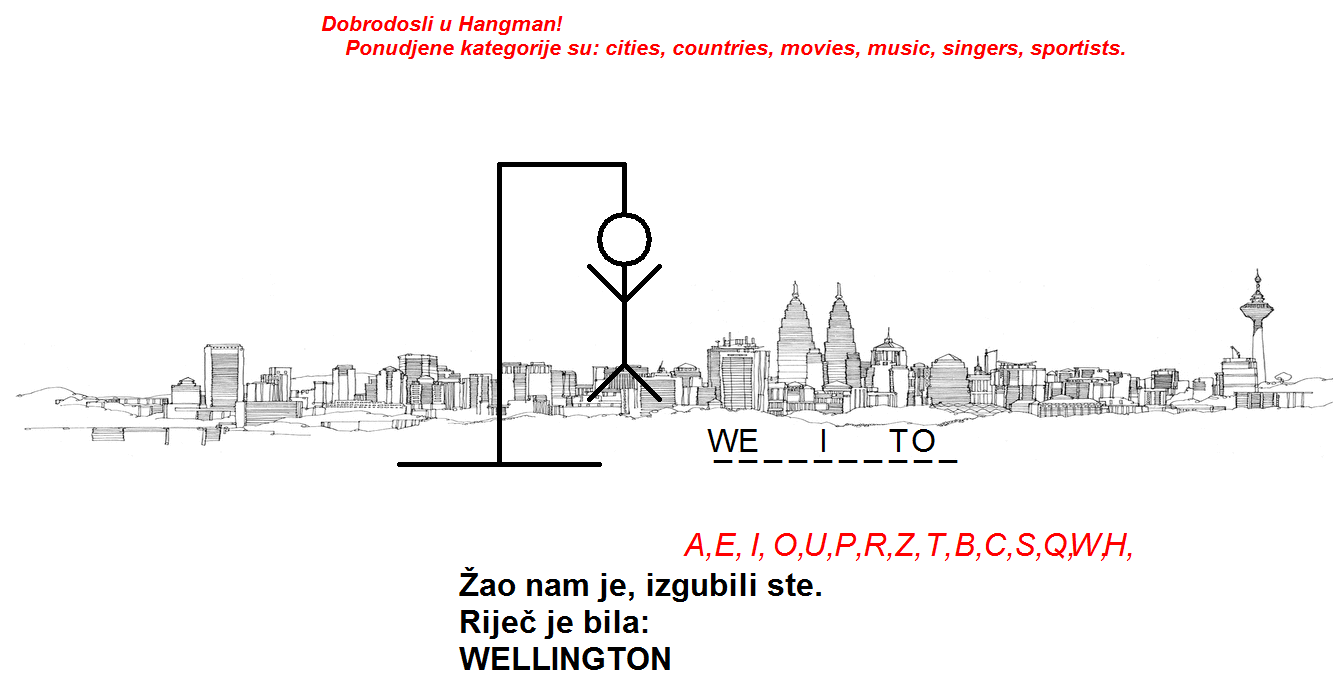


***Slika 19.****Zatvaranje tekstualnog file-a*

***4.Testiranje igrice***

Da bismo se uvjerili u tačnost i ispravnost kreiranog koda, odlučili smo se na testiranje istog. Isprobali smo kako se ponaša kada broj grešaka dodje do 10, zatim smo isporobali šta se dešava kada pobjedimo. Isprobali smo i unos riječi, tj da li i prazno mjesto uzima za riječ, te da li se vješalo kreira na ispravan način.

Prije svega testirali smo šta se dešava sa igricom ukoliko broj grešaka dođe do broja 10.Nakon što nismo uspjeli pogoditi dobivenu riječ, dobili smo poruku da smo izgubili, a zatim i rješenje.



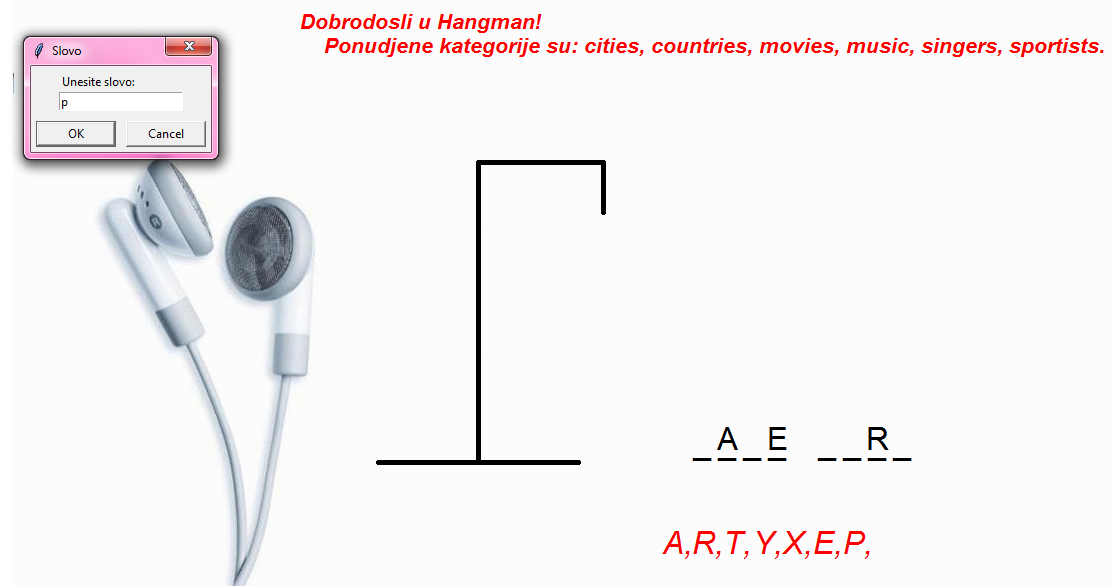
***Slika 20.****Poraz u igrici*

Nakon što smo uvidjeli da igrica radi kada broj grešaka dosegne brojku 10, željeli smo provjeriti kako program reagira kada pogodimo slova koja se nalaze u odabranoj riječi.



***Slika 21.****Pobjeda u igrici*

Zadovoljni dosadašnjim rezultatima krenuli smo u dalje testiranje. Željeli smo isprobati šta će se desiti ukoliko unesemo dva ili vise puta isto slovo. Nakon unosa istog slova program nije dozvolio ponovni unos istog, što se i trebalo dogoditi.

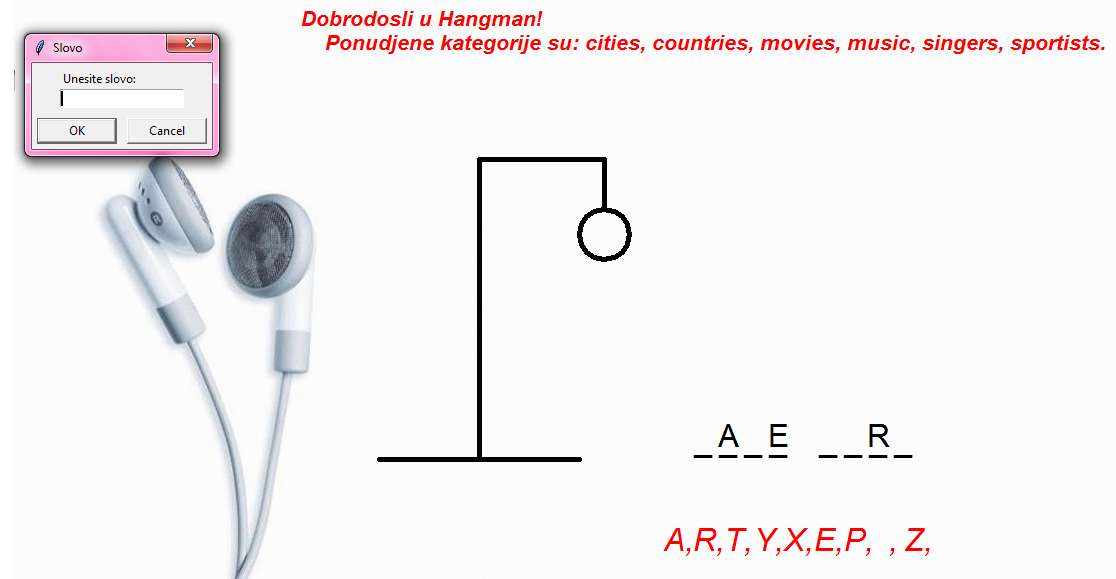


***Slika 22. a)*** *Unos već unešenog slova*



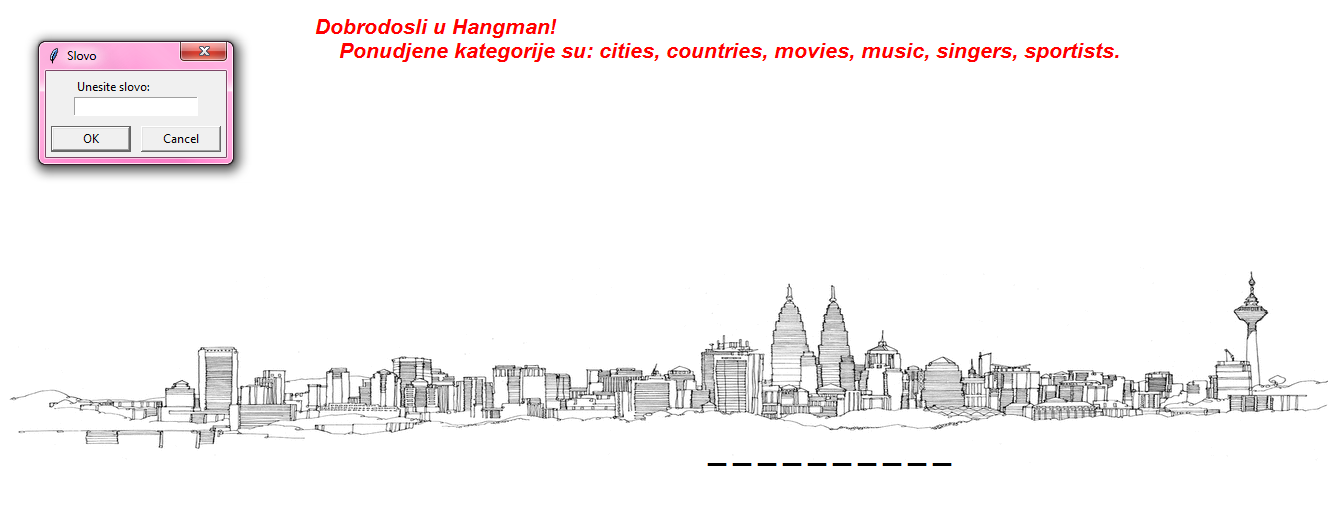
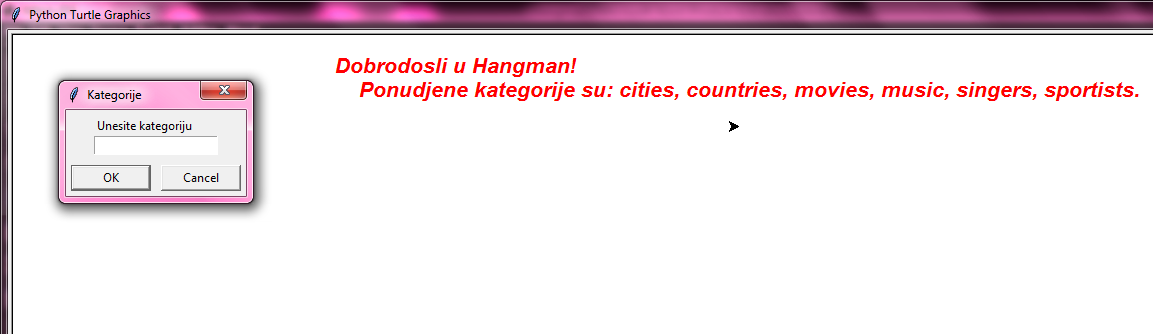
***Slika 22. b)*** *Nakon unosa već unešenog slova ponovo se pojavljuje prozorčić u koji treba da unesemo slovo*

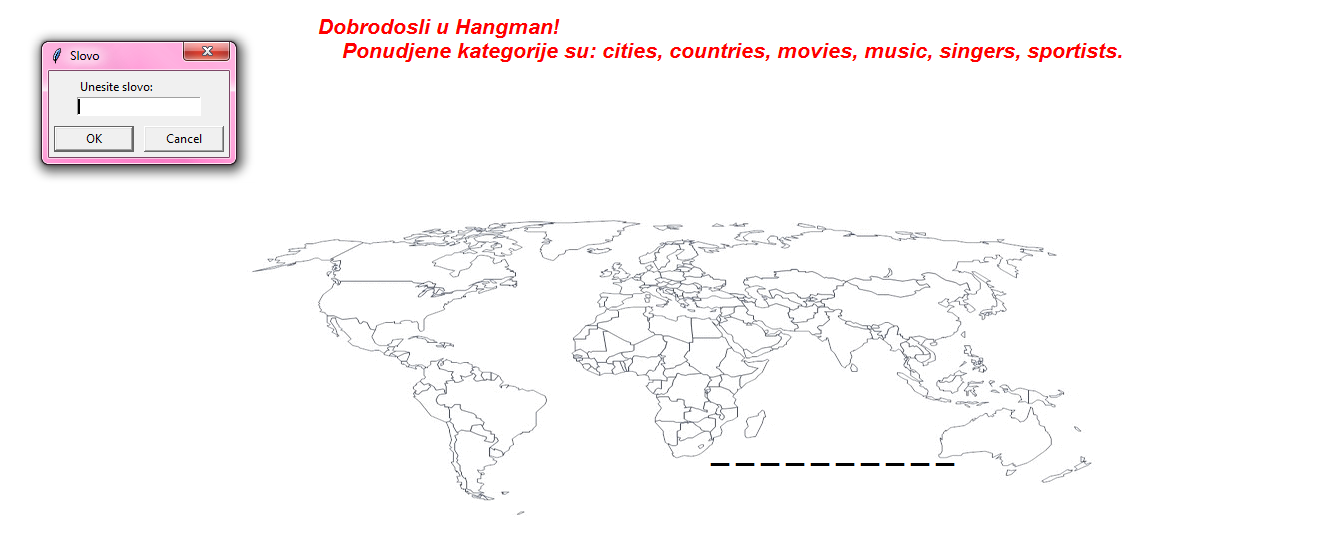
Uvidjeli smo da se sve realizira tačno onako kako smo i očekivali.Uradili smo još jedan test, kojim smo željeli uvidjeti šta se dešava ako unesemo prazno mjesto. Dogodilo se ono što smo i predvidjeli, prazno mjesto je tretirano od strane programa kao slovo, i ukoliko smo ga već jednom unjeli ne možemo to ponovo učiniti.



***Slika 23.****Unos praznog mjesta se tretira kao slovo*

Već urađenim testiranjima uvidjeli smo da se i vješalo crta na odgovarajući način. Na kraju smo zaključili, da je naša igrica dobila željeni izgled i da sve radi onako kako smo i zamislili.







***Slika 24.****Mali slikovni vodić Hangmana*

***5. Zaključak***

Radeći ovu igricu uvidijeli smo koliko zapravo programiranje može biti zanimljivo, dokučili da izlazi izvan granica monotonog i teškog,a da se perfektno uklapa u koncept zanimljivog. Upoznali smo se sa mnogim prednostima, ali i nedostacima programskog jezika Paython, te malim dijelom zavirili u svijet velikih pogramera.

Igrica koju smo odabrali, je klasik koji jako dobro kotira na ljestvicama popularnosti. Po mnogima glavni razlog tome je jednostavnost njenog koda. Nastojali smo od sasvim obične i monotone igrice dobiti nešto drugačije i inovativnije.

Rad u ovome programskom jeziku je za nas buduće programere predstavljao nešto novo. Tokom našeg ranijeg školovanja susretali smo se sa drugim programskim jezicima koji rade na drugačiji način, prvenstveno sa C++, pa smo sebi dali za pravo usporediti ova dva programska jezika.

U programskom jeziku Python-u ne postoje tzv.zamke koje su karakteristične za C++. Sama olakšica korisnicima se nalazi i u činjenici da se izbjegava unošenje znakova ***;***itd.

Na kraju bismo dodali da, sposobnost pojednostavljenja znači eliminisanje suvišnog da bi neophodno moglo progovoriti.

***6. Literatura***

1. Introduction to Computer Science Using Python: A Computional Problem-Solving Focus, Charles Dierbach