SVEUČILIŠTE U SPLITU

Sveučilišni odjel za stručne studije

OBJEKTNO ORIJENTIRANO

PROGRAMIRANJE

### Seminarski rad –

Case study: Usporedba mainstream programskih jezika(Python i C++)

Duje Budiša

Split, siječanj 2025.

### **Sadržaj**

[1. UVOD 1](#_Toc8977)

[2. RAZLIKE UPRAVLJANJA MEMORIJOM........................................................................2](#_Toc8980)

[2.1. Načini upravljanja memorijom ……………………………………………………………………………..... 3](#_Toc8980)

[2.2. Konstruktori i destruktori ………………………………………………………………………………...…….. 4](#_Toc8980)

[3. FUNKCIJE UNUTAR KLASA..........................................................................................4](#_Toc8981)

3.1Nasljeđivanje.............................................................................................................6

3.2Overriding..................................................................................................................8

3.3Shadowing...............................................................................................................10

3.4Višestruko nasljeđivanje..........................................................................................11

3.5Apstraktne klase i interface.....................................................................................15

[4. POLIMORFIZAM,GENERICI I TAMEPLATE.................................................................16](#_Toc8982)

4.1[Polimorfizam...........................................................................................................17](#_Toc8983)

4.2Vrste polimorfizma..................................................................................................19

4.3Generici i tameplate................................................................................................21

5.RAII............................................................................................................................21

6.ZAKLJUČAK................................................................................................................21

7.LITERATURA..............................................................................................................22

# UVOD

Razvoj informacijskih tehnologija unio je brojne promjene u područje programiranja, gdje su različiti programski jezici postali ključni alati za rješavanje specifičnih problema. U ovom seminarskom radu fokusirat ćemo se na usporedbu i praktičnu primjenu jezika C++(Slika1.) i Python(Slika2.) kroz analizu. C++ je jezik opće namjene poznat po svojoj brzini, učinkovitosti i snažnim mogućnostima upravljanja memorijom, što ga čini popularnim izborom za razvoj sustava, igara i aplikacija visokih performansi. S druge strane, Python se ističe svojom jednostavnošću, čitljivošću i širokom primjenom u područjima kao što su podatkovna znanost, strojno učenje i razvoj web aplikacija. Ovaj uvod postavlja temelje za dublju analizu razlika između ova dva jezika kroz konkretan primjer. Rad ima za cilj pokazati kako izbor programskog jezika može utjecati na performanse, razvoj i održavanje programskih rješenja.



*Slika1.Logo programskog jezika C++.*

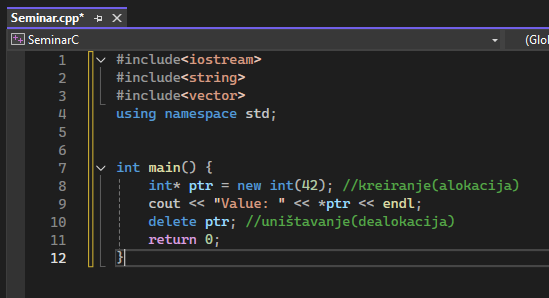


*Slika2.Logo programskog jezika Python.*

# Razlike upravljanja memorijom

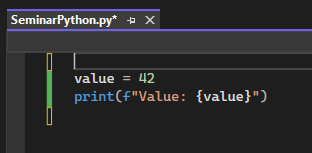
2.1 Načini upravljanja memorijom

Jedna od ključnih razlika između C++ i Pythona je način na koji se upravlja memorijom. C++ programerima omogućuje izravnu kontrolu nad memorijom putem dodjele i poništavanja koristeći ključne riječi kao što su new i delete(Slika3.). Ovo pruža visoku razinu fleksibilnosti, ali u isto vrijeme zahtijeva dodatnu pažnju kako bi se izbjegli problemi kao što su curenje memorije ili dereferenciranje nevažećih pokazivača.



*Slika3.primjer alokacije i dealokacije u C++ programskom jeziku.*

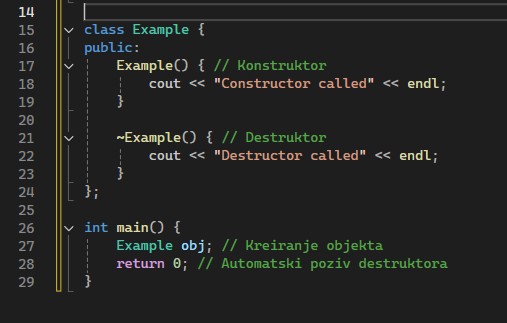
Python, s druge strane, koristi automatsko upravljanje memorijom putem mehanizama kao što je sakupljanje smeća. Ovaj pristup oslobađa programere od potrebe za ručnim upravljanjem memorijom, što pojednostavljuje razvoj i smanjuje rizik od pogrešaka, ali može utjecati na performanse u aplikacijama koje zahtijevaju visoku učinkovitost(Slika4.).



*Slika4. primjer alokacije i dealokacije u Python programskom jeziku.*

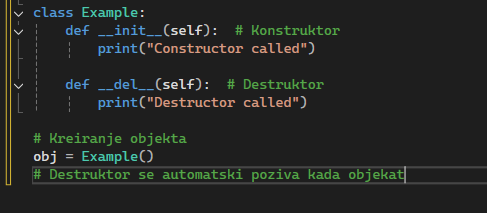
2.2 Konstruktori i destruktori

U jeziku C++, konstruktor i destruktor su posebne funkcije klase koje se automatski pozivaju prilikom stvaranja i uništavanja objekta(Slika5.). Konstruktor se koristi za inicijalizaciju objekta, dok destruktor oslobađa resurse koje koristi objekt. To omogućuje preciznu kontrolu nad životnim ciklusom objekta, što je posebno korisno u aplikacijama koje zahtijevaju upravljanje memorijom niske razine.



*Slika5. primjer konstruktora i destruktora u programskom jeziku C++.*

U Pythonu, konstruktor (\_\_init\_\_) se koristi za inicijalizaciju objekta kada je kreiran, dok se destruktor (\_\_del\_\_) koristi za čišćenje resursa kada objekt više nije potreban(Slika 6.). Međutim, zbog Pythonovog automatskog upravljanja memorijom, destruktor se rjeđe eksplicitno koristi, budući da sustav prikupljanja neiskorištenih objekata automatski oslobađa memoriju.



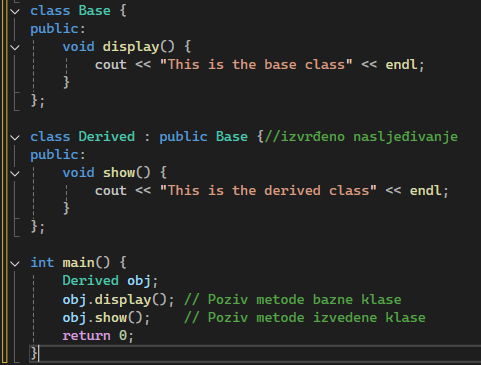
*Slika6. primjer konstruktora i destruktora u programskom jeziku Pythonu.*

# FUNKCIJE UNUTAR KLASA

3.1Nasljeđivanje

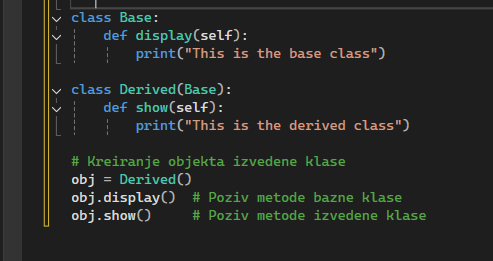
Nasljeđivanje je koncept u objektno orijentiranom programiranju koji omogućuje jednoj klasi (podklasi) da naslijedi atribute i metode druge klase (osnovne klase). To omogućuje ponovnu upotrebu koda i proširenje funkcionalnosti.

U C++ jeziku, nasleđivanje se ostvaruje pomoću sintakse class Derived : public Base(Slika7.), gdje izvedena klasa nasleđuje javne atribute i metode bazne klase.



*Slika 7. nasljeđivanje klase „Base“ u „Derived“ u C++.*

U Pythonu se nasljeđivanje postiže tako što se baza klasa navodi u zagradama prilikom definiranja podklase, kao što je klasa Derived(Base)(Slika8.). Nasljeđivanje omogućuje Python programerima da jednostavno proširi funkcionalnost postojećih klasa.

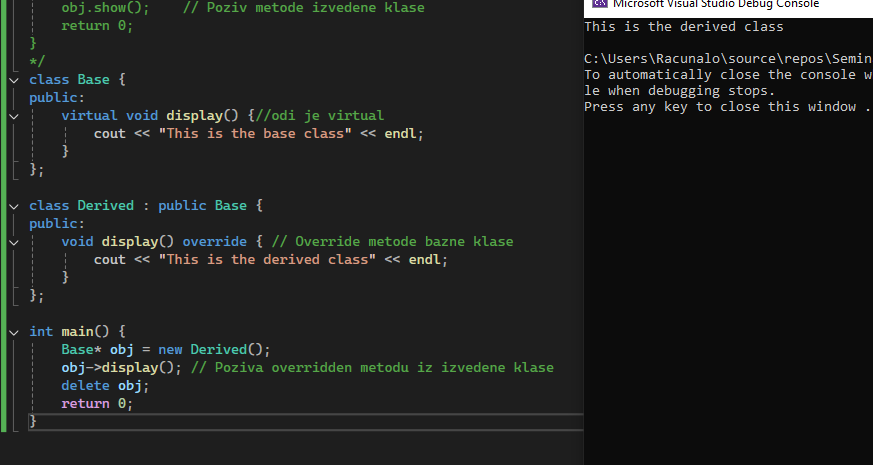


*Slika8. nasljeđivanje klase „Base“ u „Derived“ u Pythonu.*

3.2Overriding

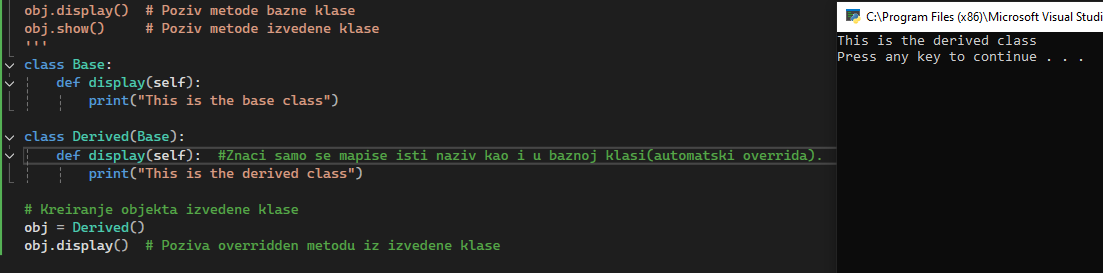
Overriding je proces u kojem podklasu definira metodu s istim imenom i potpisom kao metoda u baznoj klasi, čime zamjenjuje ponašanje bazne metode.

U jeziku C++, ključna riječ virtual se koristi u baznoj klasi da bi se omogućilo override metode u izvedenim klasama. Ključna riječ override(Slika9.) se koristi u izvedenoj klasi radi jasnoće.



*Slika9. overridanje funkcije display koristeci kljucnu rijec „override“ u C++.*

U Pythonu se overriding postiže jednostavnim definiranjem metode s istim imenom(Slika10.) u podklasi. Python automatski koristi metodu iz podklase kada je objekt izvedene klase.

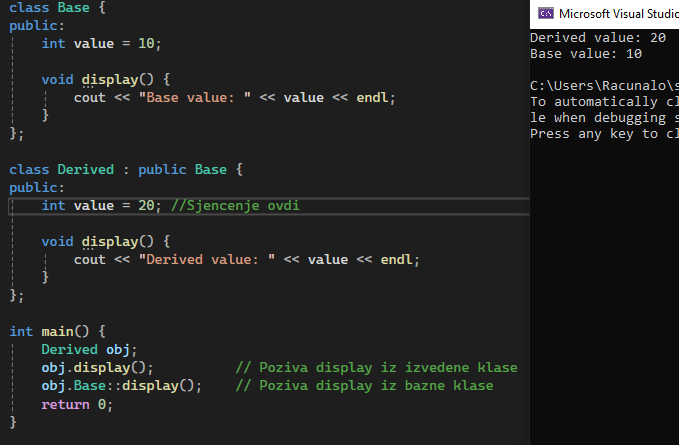


*Slika10. overridanje funkcije display koristeci isti naziv funkcije bazne klase uPythonu.*

3.3Shadowing

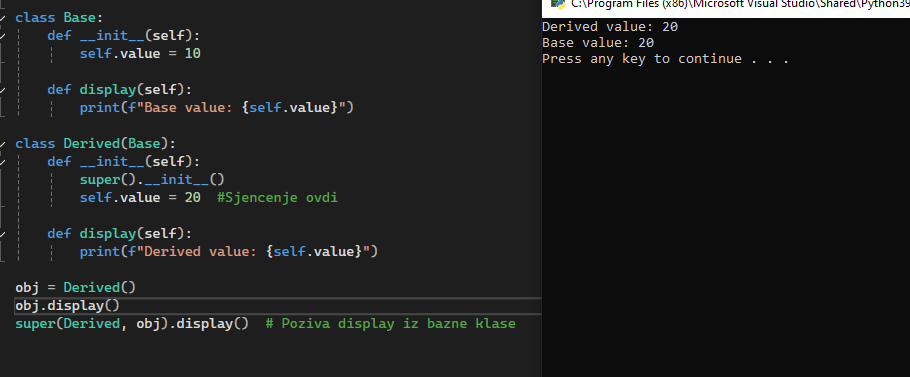
Sjenčanje (zasjenjivanje) je proces u kojem podklasu definira atribut ili metodu s istim imenom kao u baznoj klasi, ali ne zamjenjuje ponašanje bazne klase, već ga "zasenjuje".

U jeziku C++, sjenčanje omogućuje izvedenoj klasi da redefinira atribut ili metodu osnovne klase. Međutim, ponašanje osnovne klase još uvijek se može eksplicitno pozvati korištenjem sintakse Base::(Slika11.).



*Slika11.sjencenje vrijednosti(„value“)u C++.*

U Pythonu, sjenčanje se postiže redefiniranjem atributa ili metoda u izvedenoj klasi. Ponašanje osnovne klase može se eksplicitno pozvati pomoću funkcije super()(Slika12.).

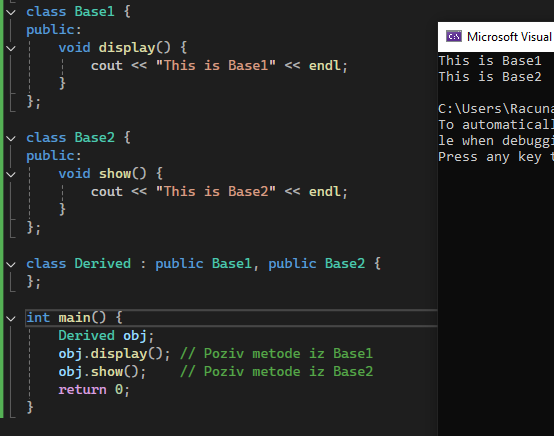


*Slika12.sjencenje vrijednosti(„value“)koristeći funkciju super u Pythonu.*

3.4 višestruko nasljeđivanje

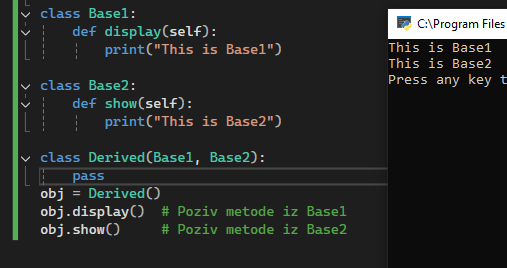
Kao i nasljeđivanje tako i višestruko nasljeđivanje omogućuje klasi da naslijedi atribute i metode iz više osnovnih klasa.

U jeziku C++ višestruko nasljeđivanje se postiže navođenjem više baznih klasa u definiciji izvedene klase(Slika13.).



*Slika13.višestruko nasljeđivanje u C++.*

U Pythonu se višestruko nasljeđivanje postiže navođenjem više baznih klasa u zagradama prilikom definiranja podklase(Slika14)..

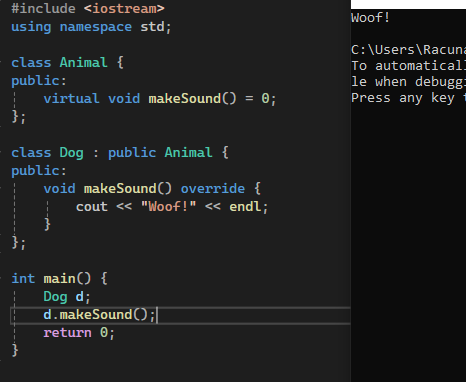


*Slika14.višestruko nasljeđivanje u Pythonu.*

3.5Apstraktne klase i Interface

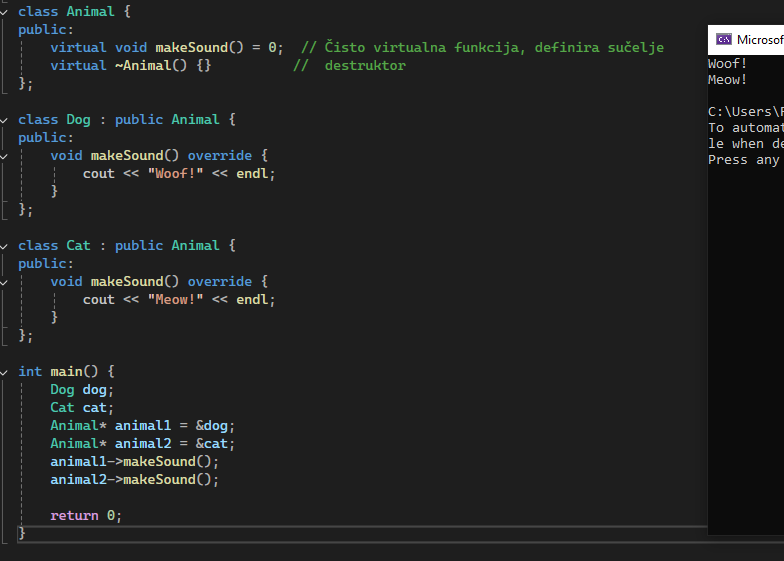
Apstraktne klase i sučelja koriste se u objektno orijentiranim programskim jezicima kao načini definiranja strukture i obveza za klase.

U C++-u, apstraktna klasa je klasa koja sadrži barem jednu čisto virtualnu funkciju.Apstraktne klase ne mogu se izravno instancirati. Naslijeđene klase moraju implementirati sve čiste virtualne funkcije ili će postati apstraktne(Slika15.).



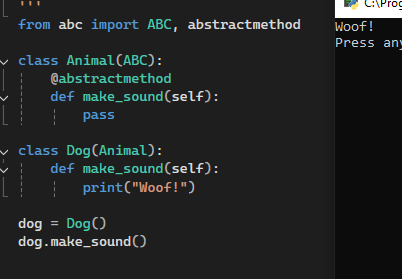
*Slika15. apstraktna klas sa čisto virtualnom funkcijom „makeSound“ u C++.*

U C++-u, sučelja se obično implementiraju kroz apstraktne klase koje sadrže samo čiste virtualne funkcije. C++ nema poseban izraz ključne riječi za sučelje kao drugi jezici.Sučelja se često koriste kao "ugovori" koje klase moraju implementirati(Slika16.).



*Slika16. sučelje animal koje nasljeđuju klase „Dog“ i „Cat“ u C++.*

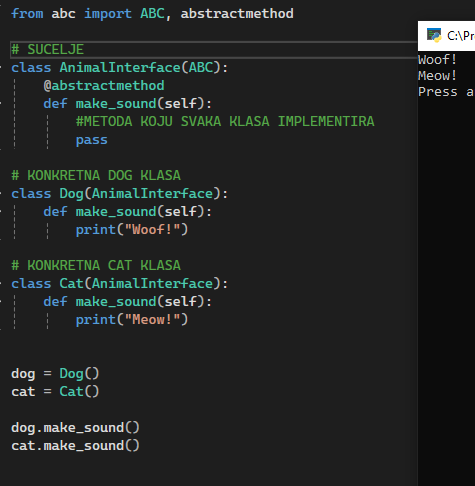
U Pythonu se apstraktne klase koriste putem abc modula i @abstractmethod dekoratora.Apstraktna klasa može imati implementirane metode, ali mora imati barem jednu apstraktnu metodu (metodu koja nema implementaciju).Naslijeđene klase moraju implementirati sve apstraktne metode da bi postale konkretne klase(Slika17.).



*Slika17. apstraktna klas koja koristi „from abc“ i „import ABC“ u PYthonu.*

U Pythonu se sučelja također mogu implementirati kroz apstraktne klase koje sadrže samo apstraktne metode.

Python ne koristi posebnu sintaksu za sučelja, ali možete definirati apstraktnu klasu koja sadrži samo apstraktne metode i tako postići sličnu funkcionalnost kao u drugim jezicima(Slika18.).



*Slika18. sučelje animal koje nasljeđuju klase „Dog“ i „Cat“ u Pythonu.*

# POLIMORFIZAM,GENERICI i TAMEPLATE

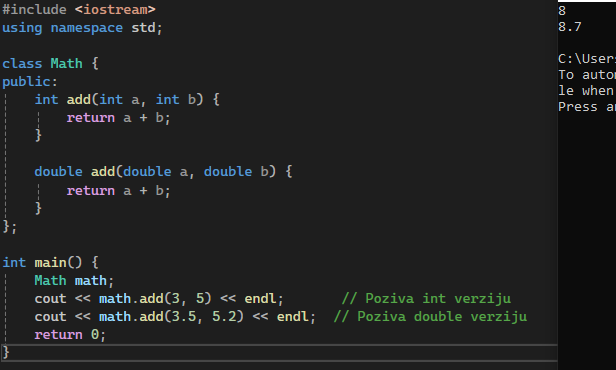
4.1Polimorfizam

Polimorfizam značajka je objektnog programiranja koja omogućuje da se objekti različitih klasa tretiraju kao objekti iste osnovne klase, zadržavajući pritom njihova specifična ponašanja. Jednostavno rečeno, polimorfizam dopušta da se ista funkcija ponaša različito ovisno o objektu koji je koristi.

4.2Vrste polimorfizama

**Compile-time** (statički polimorfizam) To se događa tijekom kompajliranja.

To se postiže preopterećenjem metode (preopterećenjem metode) ili preopterećenjem operatora(Slika19.).

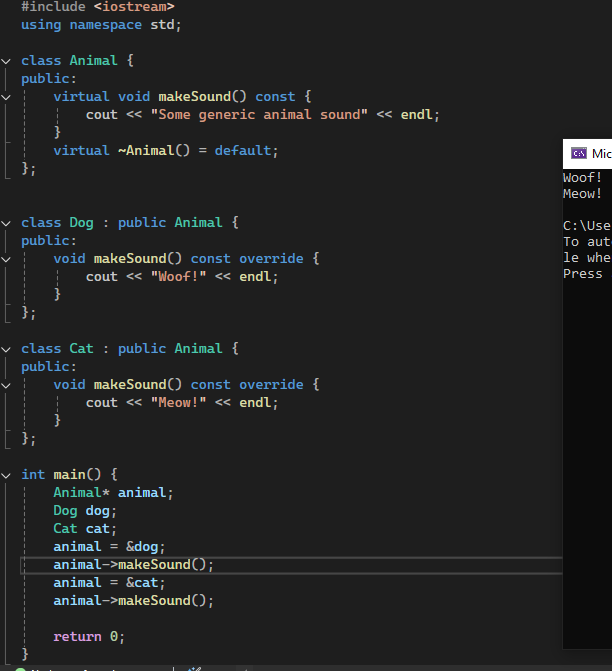


*Slika19.korištenjem metode overloadinga u C++.*

Compile-time polimorfdizam nepostoji u Pythonu jer je on dinamički tipiziran jezik, što znaći da se podaci izvršavaju u run timeu.

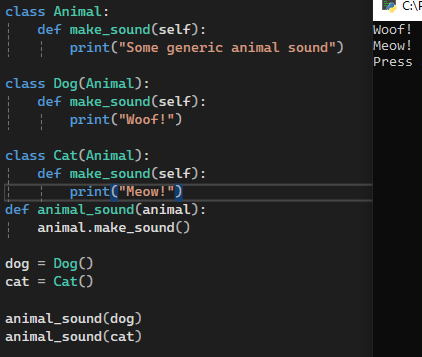
**Runtime** (ili dinamički polimorfizam) je sposobnost programa da tijekom izvođenja odluči koju će verziju funkcije pozvati, na temelju vrste objekta na koji pokazuje pokazivač ili referenca. To se postiže korištenjem virtualnih funkcija u jezicima poput C++ ili metoda u Pythonu.

U C++-u,runtime polimorfizam implementiran je pomoću virtualnih funkcija i pokazivača ili referenci na osnovnu klasu. Odluka o tome koju funkciju pozvati donosi se tijekom izvođenja na temelju stvarnog tipa objekta.U ovom primjeru su koristeni pokazivaci(Slika20.)



*Slika20.korištenjem pointera za polimorfizam u C++.*

Python prirodno podržava runtime jer je dinamički tipiziran jezik. Funkcije i metode pozivaju se na temelju stvarnog tipa objekta tijekom izvođenja, bez potrebe za dodatnim ključnim riječima kao što je „virtual“(slika21.).

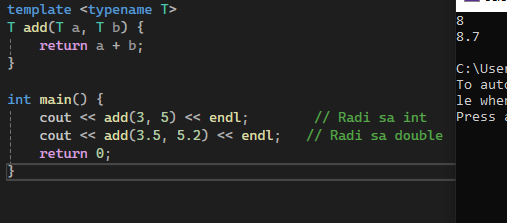


*Slika21.“prirodni“ polimorfizam u Pythonu.*

4.3Generici i tamplate

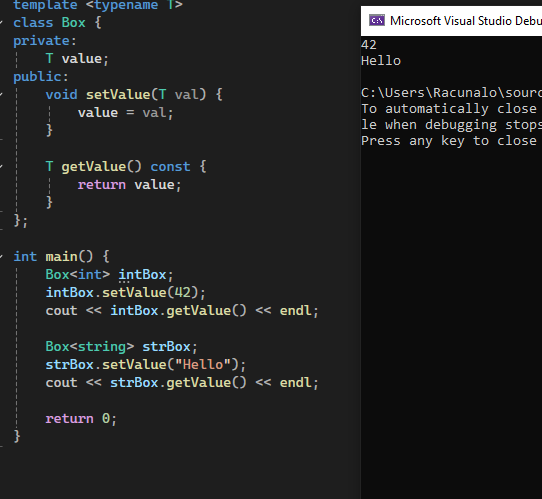
U C++-u tamplate omogućuju stvaranje funkcija i klasa koje rade s bilo kojim tipom. To se postiže korištenjem predložaka za vrijeme kompajliranja, gdje se kod generira za konkretne tipove.

Primjer tamplate-a sa funkcijom(Slika21.).



*Slika21.template kao funkcija u C++.*

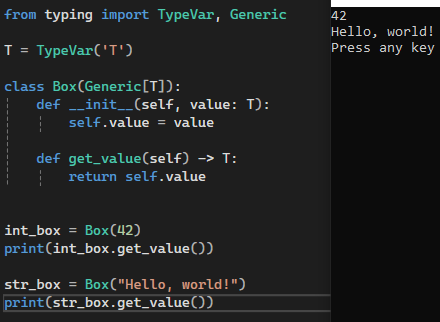
Primjer template-a sa klasom(Slika22.).



*Slika22.template korišten u klasi u C++.*

u Pythonu nema templat-ova kao u C++. Python koristi dinamičku tipizaciju, što znači da tipovi podataka nisu strogo definirani i ne provjeravaju se tijekom kompajliranja.

**generički tipovi** omogućavaju pisanje fleksibilnog koda koji može raditi s različitim tipovima podataka bez potrebe za ponovnim pisanjem funkcionalnosti za svaki tip.



*Slika23.korištenje „Generic“ unutar „Box“ u Pythonu.*

# RAII

RAII u C++ je učinkovit način upravljanja resursima jer se resursi automatski oslobađaju kada objekt izađe iz opsega, čime se smanjuje mogućnost pogrešaka uzrokovanih neobjavljenim resursima.

Python ne koristi izravno RAII, ali koristi **context manager-e** (with blok) za postizanje slične funkcionalnosti za automatsko upravljanje resursima. Python se oslanja na skupljač smeća(„Trash colector“) za upravljanje memorijom, dok RAII u C++ također uključuje resurse poput datoteka, mrežnih veza itd...

# ZAKLJUČAK

C++ i Python dva su vrlo moćna programska jezika, ali se značajno razlikuju u načinu rada, sintaksi, filozofiji dizajna i implementaciji. C++ je brži i daje potpunu kontrolu nad resursima, ali dolazi s većom složenošću u sintaksi, upravljanju memorijom i razvoju.

Python je lakši za naučiti, fleksibilniji je, ali često sporiji i manje prikladan za aplikacije koje zahtijevaju visoku izvedbu ili nisku latenciju.

Oba jezika imaju svoje specifične prednosti, a izbor između njih ovisi o specifičnim zahtjevima aplikacije i razvojnim ciljevima.

# LITERATURA

<https://www.geeksforgeeks.org/>

<https://www.geeksforgeeks.org/difference-between-python-and-c/>

<https://www.simplilearn.com/tutorials/cpp-tutorial/cpp-vs-python>

<https://www.reddit.com/r/learnpython/comments/vvhxg8/how_different_is_python_from_c/>

https://www.ionos.com/digitalguide/websites/web-development/python-vs-c/