1. **CGI interfejs**
   * CGI (Common Gateway Interface) interfejs defineše kako se informacije prenose od web servera do izvršnog programa i obrnuto. Kada korisnik traži stranicu, server mu je šalje natrag. U slučaju da korisnik popunjava neki obrazac, on se šalje serveru na obradu i posle server šalje odgovor. Ovaj metod prenosa informacija zove se CGI.
2. **Perl editori**
   * *Padre* – idealan za početnike Perl-a, ima podršku za multijezičke I multitehnološke projekte. Besplatan je, ima mogućnost provere sintakse I označavanja grešaka, radi na Windows-u, Linuxu I Mac OS-u.
   * *Ultraedit editor* – Editor namenjen za kodiranje u više jezika, podržava Perl. Od prednosti ima sve kao i Padre, dostupan je na istim platformama. Mana je što se plaća i što ima trial verzija.
   * *Epic*  - Ovo je integracija za eklipse kako bi omogućio podršku za Perl. Open source je, besplatan, podržan na sva tri operativna sistema navedena gore. Od prednosti ima sve što I prethodna dva uz ugrađenu dokumentaciju za Perl funkcije, formatiranje teksta kao I podršku za kreiranje templejta, biblioteka koda kako bi se kasnije koristili.
   * *Komodo IDE* – Podržava više scripting jezika. Dostupan je na sva tri OS-a, ima podršku za Git, Debugging I testiranje, mogućnost proširenja add-on-ovima. Mana je što nije besplatan, postoji trial verzija.
3. **Perl moduli**
   * Perl modul je pojedinačna komponenta za Perl. To je skup pravila za korišćenje mehanizma perl packages. Definiše sovj izvorni kod koji treba da se nalazi u pakovanju, ekvivalentan je klasama u OOP. Može se napraviti neki modul koji će se kasnije koristiti u različitim situacijama.
4. **Da li je Perl interpretirani ili prevođeni jezik?**
   * Zavisi iz kog ugla se posmatra I koja verzija Perla se koristi. Najnoviji Perl 6 može se smatrati kompajlianim jezikom, dok ostale verzije, starije, mogu biti smatrane interpreterskim.
5. **Tipovi Perl varijabli I domen varijabli**
   * Varijable se definišu simbolima $, @, % (jednostavne varijable, nizovi, heševi). Imena varijabli mogu sadržati alfanumeričke karaktere, mogu se koristiti mala i velika slova kao i donja crta. Ne mogu počinjati brojem. Varijable su po definiciji globalne, Da bi bila lokalna koristi se my oznaka, Primer my $var;
6. **Korišćenje dvostrukih I jednostrukih navodnih znakova u Perl-u**
   * Dvostruki navodnici I jednostruki se koriste za stringove. Dvostruki mogu uključiti unutar njih I imena drugih varijabli, varijable će biti interpolirane.
7. **Specijalni karakteri u Perl-u**
   * Specijalni karakteri koji su dostupni I koji se koriste u sklopu stringa su \n – novi red, \r – return, \t – tab, \f – formfeed, \b – backspace, \a – bell, \e – escape, \007 ili bilo koji oktalni ASCII, \x7f ili bilo koji drugi hex.

CS324 – P03

Milanka Bjelić 2626 SE

Pitanja:

1. Perl nizovi
2. Kontekst varijable
3. Perl operatori
4. Perl kontrolne strukture
5. Perl petlje
6. Perl specijalne varijable

Odgovori:

1. Perl nizovi su sekvence varijabli čiji članovi mogu biti preuzeti i dodeljeni koristeći njihove indekse. Niz počinje sa @ znakom. Niz je varijabla koja čuva uređenu listu skalarnih vrednosti. Na primer: @brojevi = (1, 2, 3);

Kako bismo specificirali jedan element niza potrebno je koristiti znak $ i ime varijable (skalar) posle čega sledi par uglastih zagrada unutar kojih je indeks elementa. $broj[3] = 4;

Elementi niza se broje od 0. Negativni indeks broji unazad od zadnjeg elementa u listi (-1 se odnosi na zadnji element).

Niz može da sadrži bilo koji tip skalara i bilo koji broj elemenata koji ne mora da se zna na početku. @voce = (“banana”, “kruska”, “jagoda”, “malina”);

2. Kontekst se odnosi na način izražavanja nekog izraza ili varijable u Perlu. Dva glavna konteksta su: skalarani kontekst i konktekst liste.

Skalarne varijable se uvek ocenjuju u skalarnom kontekstu dok se nizovi i heševi ocenjuju u oba konteksta, u sklaranom kada se tretiraju kao skalari i u kontekstu liste kada se tretiraju kao nizovi i heševi.

my @newarray = @array; # kontekst liste

my $howmany = @array; # skalarni kontekst

my $howmany2 = scalar(@array); # skalarni kontekst (explicitly)

Perl pristupa različito sa istom varijablom zavisno od konteksta. Ako imamo: my @zivotinje = (“pas”,”macka”,”kornjaca”), @zivotinje je niz, ali ako se koristi u kontekstu skalara onda vraća broj elemenata u nizu.

Perl tumači izraz zavisno od konteksta u kojem se nalazi:

42 + nešto # nešto mora biti skalar

sort nešto # nešto mora biti lista

Izrazi u Perlu vraćaju vrednost koja odgovara kontekstu u kojem se nalaze kada se ime niza nađe u kontekstu u kojem se očekuje lista, vraća elemente niza, dok se u skalarnom kontekstu vraća broj elemenata niza.

Dodeljivanje vrednosti takođe može imati različite kontekste:

@list = @people; # a list of three people

$n = @people; # the number 3

3.

sort operator – uzima listu vrednosti, sortira ih prema internom redosledu sortiranja i vraća sortiranu listu.

@rocks = sort @rocks;

strelica operator (dereferentni operator) – pokazivač na varijable, pokazuje na adresu varijable i vraća vrednost koja se nalazi na toj adresi.

my $scalar = "This is a scalar";

my $scalar\_ref = \$scalar;

print "Reference: " . $scalar\_ref . "\n";

Unarni operatori: ! (logička negacija NOT), – (obavlja aritmetičku negaciju ako je operand numerički), ~ (obavlja negaciju nad bitovima, tj 1 komplement), + (koristan za odvajanje imena funkcije iz izraza koji se nalazi u zagradama koji bi inače bio interpretiran kao kompletna lista funkcija argumenata), \ (kreira referencu bez obzira šta dođe posle tog znaka).

Aritmetički operatori: \*\* (eksponencijalni operator), \*, /, %.

Relacioni operatori: >, <, >=, <=, ==, !=.

Operatori jednakosti: ==, !=, <=>.

Autoinkrementiranje: ++

Autodekrementiranje: - -

Operatori dodeljivanja: +=, -=, \*=, /=, %=, \*\*=.

Logički operatori: &&, ||, !.

4. Kontrolne strukture su if/else/elsif koje izvršavaju blokove koda zavisno od toga da li je uslov zadovoljen.

if ( uslov ) {

blok koda

} else { blok koda

}

5. Perl podržava gotovo sve standardne petlje: while, until, do while, do until, for, foreach.

6. Specijalne varijable u Perlu: $\_, $., $/, $,, $#, $%, $=, $-, $~, $^, $|, $ $, $?, $`, $’, $0, $<number>, $[, $„, $;, $!, $@, $>, $), $:, $^D, $^F, $^I, $^L, $^P, $^W, $^X , $ARGV, @ARGV, @F, @Inc, %Inc.

1. Perl klase

Perl klasa je samo pakovanje, Perl ne daje nikakvu posebnu sintaksu za definicije klase. Pakovanje je prostor imena koji sadrži varijable i podprograme. Jedina razlika je da u klasi, podprogrami mogu očekivati referencu na objekat ili ime klase kao prvi argument. Sve klase nasleđuju UNIVERSAL klasu implicitno. UNIVERSAL klasa je implementirana Perl jezgrom, a nudi nekoliko default metoda, kao što su isa(), can() i VERSION().

2. Reference u Perl jeziku.

Referenca je pokazivač na drugi objekat. Postoje dve vrste referenci: simboličke i tvrde (engl. symbolic, hard). Simboličke reference omogućavaju da se referencira varijabla po imenu, koristeći vrednost druge varijable. Tvrda referenca se odnosi na stvarne podatke koji se nalaze u strukturi podataka.

3. Objekti u Perl jeziku.

Objekat je struktura podataka koja zna kojoj klasi pripada. Svaki objekat ili klasa imaju svoja obeležja/atribute i funkcije i ti objekti komuniciraju jedan sa drugim kada je to potrebno. Perl ne daje nikakvu posebnu sintaksu za izgradnju objekta. Objekti su samo Perl strukture podataka (heševi, nizovi, skalari, filehandles, itd) koji su eksplicitno povezani sa određenom klasom.

4. Perl pakovanja i moduli

Pakovanje je kolekcija koda koji živi u vlastitom prostoru imena što predstavlja imenovanu kolekciju unikatnih imena varijabli. Pakovanja omogućavaju izgradnju modula koji, kada se koriste, neće izmešati varijable i funkcije izvan vlastitog prostora imena nekog modula. Perl modul je pakovanje za višestruku upotrebu, definisan u biblioteci datoteke čije je ime isto kao i ime pakovanja.

5. Metode inicijalizacije

Instanciranje klase vrsi se pozivanjem metode new(); Funkcija new () je statička metoda. Statičke metode nisu povezane sa bilo kojim određenim objektom. To ima smisla jer je funkcija new () dizajnirana za stvaranje objekata. Ne može biti povezana sa bilo kojim objektom koji još ne postoji.

6. Nasledjivanje u Perlu

Nasleđivanje znači dozvolu da jedna klasa koja se zove Child nasledi metode i atribute iz druge klase koja se zove Parent, tako da ne morate iznova pisati isti kod. Na primer, možemo imati klasu Employee koja nasleđuje metode i atribute od klase Person. To se naziva "isa" odnos, između zaposleni i osoba. Perl ima posebnu varijablu, @ISA, da pomogne ovakav pristup. @ISA upravlja (metodom) nasledstva.

7. Polimorfizam u jeziku Perl

Polimorfizam je jednostavan koncept. To znači da će metode definisane u osnovnoj klasi nadjačati metode definisane u klasama roditelja.

package A;

sub foo {

print("Inside A::foo\n");

}

package B;

@ISA = (A);

sub foo {

print("Inside B::foo\n");

}

1. Definisanje I pozivanje Python funkcija

Evo nekih pravila za definisanje Python funkcija

• Blok funkcije počinje s klučnom rečju def, sledi ime funkcije i zagrade (( )). Bilo koji ulazni parametri ili argumenti trebaju biti smešteni u okviru tih zagrada. Takođe se mogu definisati parametri u okviru tih zagrada.

• Blok koda u okviru svake funkcije počinje s dvotačkom (:). Zahtev za izvršavanje funkcije se zove poziv funkcije (engl. function call). Kada se funkcija pozove, može primiti argumente koji definišu tip podataka koje funkcija koristi.

• Python funkcija uvek vraća vrednost koja može biti None ili vrednost koja predstavlja rezultat nekog proračuna izvršenog u samoj funkciji.

• Python funkcije su objekti i na njih se primjenjuju ista pravila kao i za sve druge objekte.

• Neka Python funkcija se može proslediti kao argument u pozivu neke druge funkcije.

Kod poziva funkcije navodi se ime funkcije i prosleđuju parametri, važan je redosled, broj i tip parametara u odnosu na definiciju same funkcije.

1. Argumenti python funkcija

Zadati argument - argument koji pretpostavlja zadatu vrednost ako vrednost za taj argument nije obezbeđena u pozivu funkcije. Možda će postojati potreba za više argumenata koje su potrebne funkciji ali koji nisu navedeni pri definisanju funkcije. Ovi argumenti se nazivaju argumenti promenljive dužine i oni nisu imenovani u definiciji funkcije, za razliku od potrebnih i zadatih argumenata.

1. Anonimne funkcije

Anonimne funkcije ne kreiraju se uz pomoć ključne reči def već lambda. To su male neimenovane funkcije koje mogu imati proizvoljan broj ulaznih parametara, ali samo jednu izlaznu vrednost u formi izraza. Ove funkcije se nazivaju anonimne, jer nisu definisane na standardan način pomoću def ključne reči.

1. Opseg varijabli u Pythonu

Postoje dva osnovna opsega varijabli u Pythonu:

• globalne varijable

• lokalne varijable

Varijable koje su definisane u telu funkcije imaju lokalni opseg, a one definisane van funkcije imaju globalni opseg (engl. global scope).

1. Rekurzivne funkcije u Pythonu

Rekurzivna funkcija poziva samu sebe direktno ili indirektno obično u petlji. Sumiranje sa rekurzivnom funkcijom.

1. Upravljanje izuzecima u jeziku Python

except ExceptionI: Ako postoji ExceptionI, onda izvršiti ovaj blok. except ExceptionII: Ako postoji ExceptionII, onda izvršiti ovaj blok.. ...................... else: Ako ne postoji izuzetak, onda izvršiti ovaj blok. Evo nekoliko važnih stvari o navedenoj sintaksi.

• Jedan try može imati više except izjava. Ovo je korisno kada try blok sadrži izjave koje mogu aktivirati različite vrste izuzetaka.

• Možete obezbediti generičku except klauzulu koja upravlja bilo kojim izuzetkom.

• Nakon except klauzule, može se uključiti i else klauzula. Kod u else -bloku se izvršava ako kod u try: bloku ne aktivira izuzetak.

• Else-blok je dobro mesto za kod koji ne treba zaštitu try: blokom.

# L08 – OOP u jeziku Python

1. Python klase

class ClassName:

<kod>

Definicije klase moraju biti izvršene pre nego što imaju efekat. U praksi, izjave unutar definicije klase su obično definicije funkcija, ali su i druge izjave dozvoljene, a ponekad i korisne. Definicije funkcija unutar klase obično imaju neobičnu formu liste argumenata, što je diktirano konvencijama za poziv za metode. Kada se ulazi u definiciju klase, stvoren je novi prostor imena, i koristi se kao lokalni opseg - dakle, sve dodele lokalnim varijablama ulaze u ovaj novi prostor imena.

1. Konstruktor klase u jeziku Python.

Metode su definisane na isti način kao bilo koje druge funkcije, sa napomenom da u okviru definicije klase svaki metod sadrži parametar self. Konstruktorima se može pristupiti jedino kao metodama objekta klase. \_\_init\_\_ metod je specijalni metod koji definiše konstruktor klase. Njegova odgovornost je da podesi inicijalna stanja objekta (nije obavezno da se podešavaju inicijalna stanja objekta kroz konstruktor).

1. Unistavanje objekata u jeziku Python.

Python automatski uništava nepotrebne objekte (ugrađene tipove ili instance klasa) da bi oslobodio memorijski prostor. Proces kojim Python povremeno vraća blokove memorije koji više nisu u upotrebi zove se garbage collection.

1. Nasledjivanje klasa u jeziku Python.

Umesto da se počne od početka, može se stvoriti klasa klasu koja potiče iz postojeće klase. Python obezbeđuje višestruko nasleđivanje. Nasleđena klasa sadrži sve promenljive i metode roditeljskih klasa. Nasleđene klase može menjati i onda dolazi do nadjačavanja (override) metoda roditeljske klase. Dete klasa nasleđuje atribute svog matičnog roditelja i oni se mogu koristiti kao da su definisani u klasi deteta. Klasa dete može redefinisati članove podatke i metode roditelja.

1. Python moduli.

Možemo kreirati sopstvene module, tako da možemo smeštati definicije klasa u odvojenim datotekama. Ovo eliminiše zbrku i omogućava jenostavnije ponovno korišćenje objekata u kasnijim programima. Ne mora se raditi ništa posebno da bi se kreirao modul, jedino treba staviti kod u odvojenu datoteku. Kada želimo da koristimo ovaj kod, jednostavno je potrebno koristiti import izjavu.

# Deveta lekcija – Primena jezika Python u script programiranju

1. Python Input/Output funkcije

Najjednostavniji način za generisanje izlaza je korišćenje print izjave. Za čitanje sa tastature postoje dve funkcije, input I raw\_input. Funkcija raw\_input() čita jedan red sa standardnog ulaza i vraća ga kao string (otklanja “trailing newline”).

str = raw\_input("Enter your input: ");

print "Received input is : ", str

Input je ekvivalentna sa funkcijom raw\_input osim što pretpostavlja da je ulaz validan izraz.

1. Rad sa datotekama u jeziku Python

Kao I/O postoji file system za rad sa datotekama. Pyfilesystem je Python modul koji pruža pojednostavljeni zajednički interfejs za mnoge vrste fajl Sistema.

file object = open(file\_name [, access\_mode][, buffering])

access\_mode - Određuje način rada u kojem se datoteka otvara, tj, R(Read), Rb(ReadBinary), r+(čitanje I pisanje), rb+(isto al binarno), W(write), Wb, wb+, w+, isto I sa A(append). Rename() I delete() – menjanje nayiva I brisanje datoteka.

1. Python direktorijum

Modul os ima nekoliko metoda koji pomažu kod kreiranja, preimenovanja i promene radnog direktorijuma. mkdir() - ovaj metod modula os kreira direktorijum u tekućem direktorijumu. Treba proslediti argument ovog metoda koji sadrži ime direktorijuma koji se kreira. Sintaksa os.mkdir("newdir"). Metod chdir() uzima jedan argument koji predstavlja ime direktorijuma na koji se promena dir odnosi. Sintaksa os.chdir("newdir").Metod getcwd() - ovaj metod prikazuje trenutni, radni direktorijum. Sintaksa os.getcwd(). Metod rmdir() - ovaj metod briše dati direktorijum. Pre brisanja direktorijuma, treba izbrisati datoteke u tom direktorijumu. Sintaksa os.rmdir('dirname')

1. Python baze podataka

MySQLdb je interfejs za povezivanje sa MySQL serverom baza podataka od strane jezika Python. DB API omogućava minimalni standard za rad sa bazama podataka, koristeći strukture i Python sintakse. API uključuje sledeće:

• importovanje api modula

• uspostavljanje konekcije sa bazom podataka

• izvršenje sql naredbi i uskladištenih procedura

• zatvaranje konekcije

1. CRUD operacije

db = MySQLdb.connect("localhost","testuser","test123","TESTDB" ) -> za konekciju

cursor = db.cursor() -> kreiranje cursor objekta koji se kasnije koristi ya prikupljanje podataka. Sa cursor.execute izvršavaju se sve sql naredbe(INSERT, UPDATE, DELETE,CREATE...)

Fetch

* Fetchone(): ovaj metod kupi sledeći red rezultujućeg seta upita. Rezultujući set je objekat koji je vraćen kada je objekat kursor korišćen za upit na tabeli.
* Fetchall(): ovaj metod kupi sve redove iz rezultujućeg seta
* Rowcount: ovo je read-only atribut i vraća broj redova koji su afektovani putem execute() metoda

1. Transakcije i četiri svojstva (ESID) ACID ???

Transakcije predstavljaju mehanizam koji obezbeđuje konzistentnost podataka. Transakcije bi trebalo da imaju sledeća četiri svojstva:

• atomičnost: ili se transakcija kompletira ili se ništa ne dešava

• konzistentnost: transakcija mora početi u konzistentnom stanju i mora napustiti sistem u konzistentnom stanju

• izolacija: središnji rezultati transakcije nisu vidljivi izvan te transakcije

• trajnost: jednom kada se transakcija potvrdi, efekti su stalni, čak i ukoliko sistem padne. python db api 2.0 omogućava dva metoda: ili commit ili rollback transakcije

Lekcija 10 - Uvod u jezik Ruby

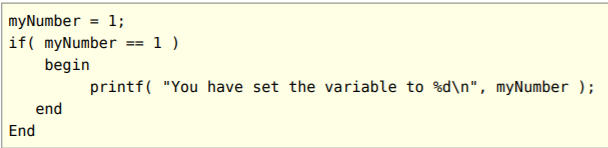
1. U kojim oblastima se Ruby dobro snalazi i zašto?

Ruby se dobro snalazi u sledećim oblastima:

* Procesiranje teksta: Ruby ima moćne alate za pronalaženje obrazaca , skeniranje stringova, zamenu, serijalizaciju podataka i različite manipulacije tekstom
* Administrativno skriptovanje: Ruby-jeve iterativne strukture su idealne za ponavljajuće i slične zadatke. Takođe, jezik integriše proceduralni stil u objektno orijentisanu arhitekturu; ukoliko želite da napišete brz-i-prljav skript da promenite naziv grupe fajlova, možete to učiniti jednostavno, ili možete uraditi na način koji je struktuiraniji, lakši za održavanje i ponovno korišćenje.
* Generisanje koda : jedna od prvih knjiga koja se fokusirala na Ruby se zove Code Generation in Action Jack Herringtona (Manning Publications, 2003), koja koristi Ruby za primere.
* Prototipovi: Ruby omogućava relativno jednostavan i prijatan rad na kreiranju prototipova kompleksnih programa koji su kreirani za razvoj u drugim programskim jezicima.
* API omotači : omogućavaju jednostavno pakovanje postojećih biblioteka u jezike više genereacije.

1. Stil kodiranja u jeziku Ruby

Ruby nam omogućava da pišemo kod koji je sličan kao u drugim programskim jezicima, uključujući i nejasni C i/ili proceduralni stil:



U nastavku je dat vodič koji se oslanja na većinsku praksu:

* + Uvlačite red sa dva prazna mesta, ne pomoću taba
  + Ne stavljajte tačku i zarez na kraju reda
  + Koristiti donju crtu \_ (my\_number) za lokalne varijable i nazive metoda
  + Koristiti CamelCase, sa početnim velikim slovom, za nazive klasa i nazive modula (MyClass)
  + Koristiti CAP\_CASE za konstante (CONST = 52)
  + Koristiti zagrade za argumente metoda

1. Šta je Ruby gem i kako se instalira?

RubyGems predstavljaju menadžera paketa za Ruby programski jezik koji omogućava standardni format za distribuciju Ruby programa i biblioteka (u formatu koji se naziva “gem”). Gems se obično gradi od ".gemspec" datoteka, koje su YAML (engl. Yet Another Markup Language) datoteke koje sadrže informacije o gemovima. Komanda gem se koristi za izgradnju, upload, preuzimanje i instaliranje gem paketa. gem install mygem.

1. Promenljive u jeziku Ruby

Promenljive i konstante nemaju tip.



Globalnim varijablama naziv počinje sa $. One mogu biti referisane sa bilo kog mesta u okviru programa. Pre inicijalizacije, globalna promeljiva ima speijalnu vrednost nil. Promenljive instance imaju imena koja počinju sa znakom @ i njihova vidljivost je ograničena. Dva različita objekta, čak iako pripadaju istoj klasi, imaju različite vrednosti za varijable instance. Izvan objekta, varijable instance ne mogu biti promenjene, čak ni pogledane (Ruby-jeve varijable instance nisu nikad public) osim kada je to ekplicitno omogućeno od strane programera. Lokalnim varijablama naziv počinje sa mali slovom ili \_ karakterom. Lokalne promenljive nemaju nil vrednost pre inicijalizacije.

1. Ruby konstante

Konstante imaju naziv koji počinje velikim slovom. Vrednost može da im bude dodeljena najviše jedanput. Konstante mogu biti definisane u okviru klasa, ali za razliku od varijbli instance, mogu biti dostupne i izvan klase. Konstante takođe mogu biti definisane u okviru modula.

1. Kontrolne strukture u jeziku Ruby

U okviru Ruby programskog jezika postoje sledeće kontrolne strukture: if, unless, while, until, for, each, case.

1. Ruby moduli

Moduli predstavljaju način za grupisanje metoda, klasa i konstanti. Moduli donose sa sobom dve bitne prednosti:

1. Oni omogućavaju da se definišu različiti prostori sa imenima (namespace) i da se na taj način spreče konflikti u imenovanju stvari. 2.
2. Moduli omogućavaju mixing funkcionalnost – funkcionalnost koja omogućava da se moduli ubace u bilo koju klasu kao njen deo (da se “umiksuju”).

Imenovanje modula je isto kao i imenovanje klasa (prvo veliko slovo).

CS324 – Lekcija 11

1. OOP u jeziku Ruby

Ruby je potpuno objektno orijentisan programski jezik, što znači da se programski problemi rešavaju definisanjem klasa, a iz klasa nastaju pojedinačni objekti.

1. Ruby nasleđivanje

Jedan od najvažnijih koncepata u OOP je koncept nasleđivanja. Nasleđivanje omogućava da definišemo klasu pomoću druge klase, što olakšava stvaranje i održavanje aplikacije. Nasleđivanje takođe pruža priliku da se ponovo koristiti funkcionalnost koda i ubrza implementacija, ali nažalost, Ruby ne podržava višestruko nasleđivanje, ali Ruby podržava mixins. Ovde je mixin slično specijalizovanoj implementaciji višestrukog nasleđivanja u kojem se samo nasleđuje interfejs.

1. Ruby super operator

Kod pozivanja super operatora bez ikakvih argumenata, Ruby šalje poruku roditelju tekućeg objekta, u kojem traži da pozove metodu sa istim imenom kao kod metode koja poziva super. To automatski prosleđuje argumente koji su preneseni metodi iz koje se zove. Kada se poziva praznom listom argumenata - super () - ne šalje nikakve argumente metodi na višem nivou, čak i ako su argumenti preneseni tekućoj metodi.

1. Ruby singleton metod

Metod koji se dodeljuje samo jednom objektu se naziva singleton. Singleton metodi se obično koriste za elemente GUI-a, kada različita akcija treba da bude pokrenuta u slučaju da je različito dugme pritisnuto. Singleton metod nije jedinstven za Ruby, on se takođe nalazi i u CLOS, Dylan, itd.

1. Ruby moduli i mixins

Moduli u jeziku Ruby su slični klasama.

• modul ne može da ima instance

• modul ne može da ima podklase

• modul je definisan sa: module ... end.

Moduli obezbeđuju prostor imena i sprečavaju kolizije imena. Moduli implementiraju mixin termin. Moduli definišu prostor imena, "sandbox" u kojem su metode i konstante zaštićene od drugih metoda i konstanti. Ruby ne podržava direktno višestruko nasleđivanje, ali Ruby moduli imaju još jednu karakteristiku. Oni su prilično eliminisali potrebu za višestrkim nasleđivanjem, koristeći termin mixin. Mixins daje kontrolisani način za dodavanje funkcionalnosti klasama. Međutim, njihova prava moć dolazi kada kod u mixin-u počne interakciju sa kodom u klasi koju koristi.

1. Ruby izjava rescue

Izvršenje programa može da uđe u nepredviđene probleme. Datoteka koju želimo da čitamo možda ne postoji, disk može biti pun i slično. Izvršenje i izuzetak uvek idu zajedno. Ako se otvara datoteka, koja ne postoji, onda je program lošeg kvaliteta. Program se zaustavlja ako dođe izuzetka. Dakle, izuzeci se koriste za rukovanje raznim vrstama grešaka, koje se mogu pojaviti tokom izvršavanja programa. Treba preduzeti odgovarajuće akcije, umesto zaustavljanja programa u potpunosti. Ruby ima mehanizam za rukovanje izuzecima. Izuzetak se moze podici u begin / end bloku i koristiti klauzulu rescue da se kaže jeziku Ruby o vrsti izuzetka za rukovanje.