Siedem języków w siedem tygodni. Praktyczny przewodnik nauki języków programowania

Bruce A. Tate

Drogi Czytelniku! Poniżej zamieszczona jest errata do książki:
"Siedem języków w siedem tygodni. Praktyczny przewodnik nauki języków programowania"

Jest to lista błędów znalezionych po opublikowaniu książki, zgłoszonych i zaakceptowanych przez naszą redakcję. Pragniemy, aby nasze publikacje były wiarygodne i spełniały Twoje oczekiwania. Zapoznaj się z poniższą listą. Jeśli masz dodatkowe zastrzeżenia, możesz je zgłosić pod adresem https://helion.pl/user/erraty

Strona	Linia	Jest	Powinno
217	33	9> X < 0 -> negative;	9> X < 0 -> fałsz;
269	23	repl odpowiedziała, zwracając sekwencję.	rest odpowiedziała, zwracając sekwencję.

Poniżej znajduje się lista błędów znalezionych przez czytelników, ale jeszcze nie potwierdzonych przez Redakcję:

Strona	Linia	Jest	Powinno
46	**-**	**drzewo_ruby.odwiedź = {|węzeł| puts węzeł.nazwa_węzła}**	**drzewo_ruby.odwiedź {|węzeł| puts węzeł.nazwa_węzła}**
46	**drzewo_ruby.odwiedź_wszystkie -**	**= {|węzeł| puts węzeł.nazwa_węzła}**	**drzewo_ruby.odwiedź_wszystkie {|węzeł| puts węzeł.nazwa_węzła}**
48	**-**	**Moduł jest kolekcją funkcji i stałych. Jeśli włączymy model do klasy, [...]**	**Moduł jest kolekcją funkcji i stałych. Jeśli włączymy moduł do klasy, [...]**

74	10	Io> metoda() type	list(1, 2, 3, 4)
120	22	Druga klauzula jest bardziej złożona: przodek(X, Y) :- ojciec(X, Z), ojciec(Z, Y).	Druga klauzula jest bardziej złożona: przodek(X, Y) :- ojciec(X, Z), przodek(Z, Y).
158	9	batate$ scala code/scala/forLoop.scala wszystko całość składa się z kawałków	batate$ scala code/scala/forLoop.scala całość składa się z kawałków
161	26	println("Obrót " + zmieńKierunek + "Obecne nastawy " + direction)	println("Obrót " + zmieńKierunek + "Obecne nastawy " + kierunek)
169	30	scala> var zmienna_niemutowalna = "Nie jestem mutowalna"	scala> val zmienna_niemutowalna = "Nie jestem mutowalna"
184	1	scala> movies.text	scala> filmy.text
206	19	4 + "ciąg_znaków".	4 + "ciag_znakow".
206	4	"ciąg znaków".	"ciag znakow".
207	14	Pigułka = niebieska.	Pigulka = niebieska.
208	25	Osoba = {osoba, {nazwisko, "Agent Smith"}, {zawód, "Zabijanie programów"}}.	Osoba = {osoba, {nazwisko, "Agent Smith"}, {zawod, "Zabijanie programow"}}.

208	8	{komiks, {nazwa, "Calvin i Hobbes"}, {postać, "Spaceman Spiff"}}.	{komiks, {nazwa, "Calvin i Hobbes"}, {postac, "Spaceman Spiff"}}.
209	13	[Głowa | Ogon] = [1, 2, 3].	[Glowa | Ogon] = [1, 2, 3].
209	15	Głowa.	Glowa.
212	1	lustro(śmiejąca_się_gęba).	lustro(smiejaca_sie_geba).
212	3	prosty:lustro(śmiejąca_się_gęba).	prosty:lustro(smiejaca_sie_geba).
216	13	Zwierzę= "pies".	Zwierze = "pies".
216	14	case Zwierzę of	case Zwierze of
216	23	case Zwierzę of "słoń" -> dumbo; _ -> coś_innego end.	case Zwierze of "slon" -> dumbo; _ -> cos_innego end.
217	20	X = 0. if X > 0 -> prawda; X < 0 -> fałsz end.	X = 0. if X > 0 -> prawda; X < 0 -> falsz end.
217	31	if X > 0 -> prawda; X < 0 -> negative; true -> zero end.	if X > 0 -> prawda; X < 0 -> nieprawda; true -> zero end.
220	3	Wyświetl = fun(X) -> io:format("~p~n", [X]) end.	Wyswietl = fun(X) -> io:format("~p~n", [X]) end.

222	16	Sumator = fun(ElementListy, SumaCzęściowa) -> ElementListy + SumaCzęściowa end.	Sumator = fun(ElementListy, SumaCzesciowa) -> ElementListy + SumaCzesciowa end.
222	18	SumaPoczątkowa = 0. 0 lists:foldl(Sumator, SumaPoczątkowa, Liczby).	SumaPoczatkowa = 0. 0 lists:foldl(Sumator, SumaPoczatkowa, Liczby).
225	14	ZVAT = [{Produkt, Ilość, Cena, Cena Ilość 0.08} || {Produkt, Ilość, Cena} <- Koszyk].	ZVAT = [{Produkt, Ilosc, Cena, Cena Ilosc 0.08} || {Produkt, Ilosc, Cena} <- Koszyk].
225	9	Koszyk = [{ołówek, 4, 0.25}, {długopis, 1, 1.20}, {papier, 2, 0.20}].	Koszyk = [{olowek, 4, 0.25}, {dlugopis, 1, 1.20}, {papier, 2, 0.20}].

229	3	- module(przetlumacz). -export([petla/0]). petla() -> receive "house" -> io:format("dom~n"), petla(); "white" -> io:format("biały~n"), petla(); _ -> io:format("Nie rozumiem.~n"), petla() end.	- module(przetlumacz). -export([petla/0]). petla() -> receive "house" -> io:format("dom~n"), petla(); "white" -> io:format("bialy~n"), petla(); _ -> io:format("Nie rozumiem.~n"), petla() end.
233	20	Z ! "biały",	Z ! "bialy",
233	22	{Z, _} -> Z ! "Nie rozumiem." , loop()	{Z, _} -> Z ! "Nie rozumiem." , petla()
234	29	petla() -> receive 3 -> io:format("bum.~n"), exit({ruletka,die,at,erlang:time()}); _ -> io:format("kliknij~n"), loop() end.	petla() -> receive 3 -> io:format("bum.~n"), exit({ruletka,die,at,erlang:time()}); _ -> io:format("kliknij~n"), petla() end.

236	11, 30	{´EXIT´, Od, Przyczyna} -> io:format("Strzelec ~p zginął z powodu ~p." , [Od, Przyczyna]), io:format("Uruchom kolejny proces.~n"), petla() end.	{´EXIT´, Od, Przyczyna} -> io:format("Strzelec ~p zginal z powodu ~p." , [Od, Przyczyna]), io:format("Uruchom kolejny proces.~n"), petla() end.
238	14	io:format("Strzelec ~p zginął z powodu ~p." , [Od, Przyczyna]), io:format(" Restart. ~n"),	io:format("Strzelec ~p zginal z powodu ~p." , [Od, Przyczyna]), io:format(" Restart. ~n"),
249	28	(/ 2,0 4)	(/ 2.0 4)
274	34	(iterate para-fibonacci [1 1])	(iterate para-fibonacci [1 1]))
283	21	user=> (deref	user=> (deref film)
285	1	user=> (reset! zagrozenie "Podział bez ryzyka."))	user=> (reset! zagrozenie "Podział bez ryzyka.")