

knihovna programátora

- Učebnice jazyka i referenční příručka v jednom
- Podrobný výklad konstrukcí objektově orientovaného programování
- Přehledné vysvětlení základních principů, na nichž je jazyk postaven a jejichž znalost umožní lépe využívat jeho možnosti
- Dekorátory, deskriptory, generátory, metatřídy
- Vlákna, procesy, asynchronní konstrukce, subinterpret
- Navazuje na titul věnovaný algoritmickým konstrukcím



Rudolf Pecinovský

Python 3.14

OBJEKTOVÉ KONSTRUKCE



*Mé ženě Jarušce a dětem
Štěpánce, Pavlínce, Ivance a Michalovi*

knihovna programátora

RUDOLF PECINOVSKÝ

Python 3.14

OBJEKTOVÉ KONSTRUKCE

GRADA
Publishing

Rudolf Pecinovský

Python 3.14

Objektové konstrukce

Vydala Grada Publishing, a.s.
U Průhonu 22, Praha 7
obchod@grada.cz, www.grada.cz
tel.: +420 234 264 401
jako svou 10352. publikaci

Odpovědný redaktor Petr Somogyi
Grafická úprava a sazba Rudolf Pecinovský
Počet stran 600
První vydání, Praha 2025
Vytiskly Tiskárny Havlíčkův Brod, a. s.

© Grada Publishing, a.s., 2025
Cover Design © Grada Publishing, a.s., 2025
Cover Photo © Depositphotos/cristi180884

Upozornění pro čtenáře a uživatele této knihy
Všechna práva vyhrazena. Žádná část této tištěné či elektronické knihy nesmí být reprodukována a šířena v papírové, elektronické či jiné podobě bez předchozího písemného souhlasu nakladatele. Neoprávněné užití této knihy bude trestně stíháno. Názvy produktů, firem apod. použité v knize mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků. Automatizovaná analýza textů nebo dat ve smyslu čl. 4 směrnice 2019/790/EU a použití této knihy k trénování AI jsou bez souhlasu nositele práv zakázány.

ISBN 978-80-271-8362-3 (pdf)

ISBN 978-80-271-6062-4 (print)

Stručný obsah

Úvod	28
Část A Objektově orientované programování	37
Kapitola 1 Základy OOP	38
Kapitola 2 Třídy a jejich instance	59
Kapitola 3 Jednoduché dědění	90
Kapitola 4 Násobné dědění	105
Kapitola 5 Vlastnosti, abstraktní třídy a protokoly	123
Kapitola 6 Další objektové konstrukce	146
Část B Prohloubení probraných témat	173
Kapitola 7 Moduly, balíčky, virtuální prostředí	174
Kapitola 8 Volatelné objekty a jejich vlastnosti	200
Kapitola 9 Dekorátory	215
Kapitola 10 Třídní metody	229
Kapitola 11 Iterátory a generátory	241
Kapitola 12 Výčtové typy	260
Kapitola 13 Datové třídy	292
Kapitola 14 Anotace a koncept typového systému	308
Kapitola 15 Generické datové typy a generické funkce	338
Kapitola 16 Výjimky a varování	358

Část C Pokročilejší objektové konstrukce	385
Kapitola 17 Přetěžování operátorů	386
Kapitola 18 Ovlivnění přístupu k atributům	416
Kapitola 19 Správa paměti a pokročilé techniky práce s objekty	445
Kapitola 20 Ovlivnění tvorby tříd, metatřídy	456
 Část D Programování souběžných úloh	 477
Kapitola 21 Principy souběžného programování	478
Kapitola 22 Vlákna a modul threading	497
Kapitola 23 Podprocesy a modul multiprocessing.....	520
Kapitola 24 Korutiny a modul asyncio	546
Kapitola 25 Subinterprety a modul concurrent.interpreters	567
 Literatura.....	 591
Rejstřík.....	593

Podrobný obsah

Úvod	28
Komu kniha není určena	28
Komu je kniha určena	29
Struktura příručky.....	29
Koncepte výkladu	30
Jazyk identifikátorů	31
Potřebné vybavení.....	31
Operační systém.....	32
Doprovodné programy a jejich organizace.....	32
Odkazy do prvního dílu a modul dbg	33
Použité typografické konvence	33
Odbočka – podšeděný blok.....	35
Zpětná vazba	36
 Část A Objektově orientované programování	37
Kapitola 1 Základy OOP	38
1.1 Předehra	38
1.1.1 Kdy se OOP začíná vyplácet	39
1.1.2 Objektově orientované paradigma	39
Postup dle procedurálního paradigmatu	40
Postup dle objektově orientovaného paradigmatu	40
1.1.3 Výhody postupu podle OOP.....	40
1.1.4 Různé pohledy na OOP.....	41
1.2 Základní princip OOP	42
1.3 Objekty a jejich atributy	42
1.4 Třídy a jejich instance	43
Terminologická vsuvka.....	44
1.4.1 Třída versus datový typ	45
1.4.2 Instance	45
1.5 Objekt třídy versus instance třídy	46
1.6 Atributy třídy versus atributy instancí.....	46
1.7 Zprávy	47
1.8 Metody	47
1.8.1 Metody versus funkce.....	48
1.9 Dědění.....	48
1.9.1 Terminologie	48
Dědění versus dědičnost.....	49
1.9.2 LSP – substituční princip Liskové.....	49
1.9.3 Virtuální metody a jejich přebíjení	49
1.9.4 Zakrývání versus přebíjení	50
1.9.5 Polymorfismus.....	50
1.9.6 Rodičovský podobjekt	51

1.9.7	Násobné dědění a diamantový problém.....	51
1.9.8	Zobecňování.....	52
1.10	Obecnější pohled na dědění.....	53
1.10.1	Rozhraní versus implementace.....	53
1.10.2	Signatura versus kontrakt.....	54
1.10.3	Formalizace popisu rozhraní – API.....	54
1.10.4	Dva způsoby dědění.....	55
	Jmenovité dědění (nominal subtyping).....	55
	Strukturální dědění.....	55
1.10.5	Tři druhy dědění.....	56
	Přirozené (nativní) dědění.....	56
	Dědění rozhraní.....	56
	Dědění implementace.....	57
	Shrnutí.....	57
1.11	Shrnutí kapitoly a doprovodné soubory.....	58
1.11.1	Doprovodné soubory.....	58
Kapitola 2	Třídy a jejich instance.....	59
2.1	Definice třídy a jejích atributů.....	59
2.1.1	Definice třídy je příkaz.....	61
2.2	Práce s atributy objektu.....	61
2.2.1	Získání a modifikace hodnoty atributu.....	61
2.2.2	Přidání a odebrání atributu.....	62
2.2.3	Omezení zabudovaných typů.....	64
2.2.4	Neveřejné atributy.....	64
2.2.5	Třída jako parametr a návratová hodnota.....	64
2.3	Instance a práce s nimi.....	65
2.3.1	Textový podpis instance.....	66
2.3.2	Kvalifikace atributu třídy instancí.....	66
	Datové atributy.....	66
	Metody.....	67
2.3.3	Tři druhy metod a použití příslušných dekorátorů.....	68
2.3.4	Metody instancí a parametr self.....	71
2.3.5	Vytváření instancí – konstruktor, alokátor, initor.....	73
2.3.6	Instanční metody třídy C2.....	74
	Terminologická vsuvka: co je konstruktor.....	75
2.3.7	Použití instančních a třídních atributů.....	76
2.3.8	Změny instančních a třídních datových atributů.....	77
2.3.9	Atribut třídy jako výchozí hodnota instančního atributu.....	78
2.3.10	Převod metod na funkce a funkcí na metody.....	80
2.3.11	Zavádění nových třídních funkčních atributů.....	81
2.3.12	Zavádění nových instančních funkčních atributů.....	82
2.4	Vestavěné třídy jsou nemodifikovatelné.....	83
2.5	Nutnost kvalifikace atributů.....	83
2.6	Některé speciální atributy – dunderý.....	85
	__bases__.....	85
	__class__.....	85
	__dict__.....	85
	__dir__(self).....	85
	__doc__.....	85
	__module__.....	85
	__mro__.....	85
	__name__.....	86
	__qualname__.....	86
	__subclasses__().....	86
	Ukázky použití.....	86
2.7	Alternativní definice třídy.....	87

2.8 Odložené vyhodnocování anotací.....	88
2.9 Shrnutí kapitoly a doprovodné soubory.....	88
2.9.1 Doprovodné soubory	89
Kapitola 3 Jednoduché dědění.....	90
3.1 Vytváření potomka a jeho vlastnosti	90
3.2 Příklad.....	91
3.2.1 Třída LA.....	91
3.2.2 Třídy LB a LC	93
3.3 Jmenné prostory.....	93
3.3.1 Atributy a jmenné prostory tříd.....	94
3.3.2 Atributy a jmenné prostory instancí.....	95
3.4 Volání metod v hierarchii dědění	96
3.4.1 Jak potvrdit anebo zakázat přebíání metody.....	97
3.5 Initory se dědí	97
3.6 Hierarchie výjimek.....	98
3.6.1 Dopad hierarchie výjimek na jejich zpracování	99
Povinnost řadit rodiče za potomky	100
To byl jenom začátek.....	100
3.7 Definice vlastní výjimky	101
3.8 Varování a jeho hierarchie.....	102
3.9 Vestavěné funkce pracující s objekty	102
callable(object).....	102
classmethod(method, *args, **kwargs).....	102
delattr(object, name)	102
getattr(object, name[, default])	102
hasattr(object, name)	103
isinstance(object, classinfo)	103
issubclass(class, classinfo)	103
setattr(object, name, value)	103
staticmethod(method, *args, **kwargs).....	103
super([type[, object-or-type]]).....	103
class type(object) class type(name:str, bases:tuple[type], dict:dict[str,Any], **kwargs) -> type.....	103
3.10 Shrnutí kapitoly a doprovodné soubory.....	104
3.10.1 Doprovodné soubory	104
Kapitola 4 Násobné dědění.....	105
4.1 Python a násobné dědění	105
Vytváření posloupnosti MRO	106
4.2 Příklad: jednoduchý diamant.....	107
4.3 Analýza chování.....	108
4.3.1 Přímé zadání volaného initoru	110
4.3.2 Problémy s parametry a argumenty – hubnoucí parametr	111
4.3.3 Virtuální metody	113
4.4 Pořadí prohledávání – MRO	113
4.5 Složitější hierarchie dědění	116
4.5.1 Nerealizovatelná a následně opravená hierarchie	116
4.5.2 Zásady specifikace MRO	117
4.6 Komolení jmen a pseudosoukromé atributy.....	118
4.7 Šablonová metoda.....	119
4.8 Shrnutí kapitoly a doprovodné soubory.....	121
4.8.1 Doprovodné soubory	122

Kapitola 5 Vlastnosti, abstraktní třídy a protokoly	123
5.1 Atributy × vlastnosti	123
5.2 Přímá definice vlastnosti	124
5.2.1 Využití lambda-výrazů, atributy jen pro čtení	126
5.2.2 Vztah vlastnosti k třídě a instancím	128
5.3 Zadání vlastnosti pomocí dekorátoru	129
5.3.1 Použití	130
5.3.2 Shrnutí	132
5.4 Definice vlastností třídy	132
5.5 Abstraktní × konkrétní třídy a metody	132
5.5.1 Princip abstraktních metod	133
5.5.2 Omezení abstraktních tříd	133
5.5.3 Abstraktní a konkrétní třídy v mainstreamových jazycích	134
5.5.4 Formálně a skutečně abstraktní třídy a metody v jazyce Python	134
5.6 Definice skutečně abstraktní třídy	135
5.6.1 Přebíjet musí i ti, kteří abstraktní metodu nepotřebují	136
5.6.2 Definice potomka abstraktní třídy	136
5.6.3 Proč definovat metodu použitou v šabloně	137
5.6.4 Proč dekorovat metodu jako abstraktní	138
5.6.5 Proč v hlavičce deklarovat předka ABC	138
5.6.6 Shrnutí	138
5.7 Abstraktní statické a třídí metody a vlastnosti	138
5.8 Statické, dynamické a kachní typování	140
5.8.1 Statické typování	140
5.8.2 Dynamické typování	140
5.8.3 Kachní typování	140
5.8.4 Statické kachní typování a jeho kontrola	141
5.9 Protokoly	142
5.9.1 Deklarace rozhraní v mainstreamových jazycích	142
5.9.2 Třída Protocol a dekorátor runtime_checkable	142
5.9.3 Protokoly a statické kachní typování	145
5.10 Shrnutí kapitoly a doprovodné soubory	145
5.10.1 Doprovodné soubory	145
Kapitola 6 Další objektové konstrukce	146
6.1 Výčtové typy – základy	146
6.1.1 Změny mezi verzemi	147
6.1.2 Trocha terminologie	147
Implicitní podoba podpisů	147
6.1.3 Výčtový typ definovaný voláním funkce	147
6.1.4 Vlastnosti výčtového typu a jeho instancí	149
Tři způsoby získání instance výčtového typu	149
Další zajímavé informace	149
6.1.5 Výčtový typ definovaný jako třída	150
6.1.6 Neměnné a proměnné části	151
6.2 Datové třídy – základy	152
6.3 Generické datové typy	154
6.4 Srovnávání objektů se vzory v příkazu match	154
6.4.1 Opakování terminologie	154
6.4.2 Dosazování hodnot	155
6.4.3 Prověřování posloupností	156
6.4.4 Vzory s výběrem hodnot	158
6.5 Definice vlastního správce kontextu	159
__exit__(self, exc_type, exc_value, traceback)	160
6.5.1 Uschopnění robota Karla vystupovat jako správce kontextu	160
6.5.2 Příklad využívající robota Karla jako správce kontextu	162
6.5.3 Sdužování	163

6.5.4 Příklad.....	163
6.5.5 Jak to funguje – emulace příkazu with	163
6.6 T-stringy – bezpečné šablony	166
6.6.1 Syntaxe a vytvoření šablony.....	166
6.6.2 Princip fungování t-stringů	167
6.6.3 API t-stringů.....	167
Atributy instancí třídy <code>string.Template</code>	167
Atributy instancí třídy <code>string.Template.Interpolation</code>	168
6.6.4 Zobrazení výsledného stringu	168
Applikace na HTML	170
6.6.5 Dosazování nových hodnot.....	170
6.6.6 Porovnání t-stringů s f-stringy a třídou <code>string.Template</code>	171
6.7 Shrnutí kapitoly a doprovodné soubory.....	172
6.7.1 Doprovodné soubory	172

Část B Prohloubení probraných témat 173

Kapitola 7 Moduly, balíčky, virtuální prostředí	174
7.1 Zprostředkovaný import	174
7.2 Oprava načteného modulu	175
7.2.1 Korektní opětovné načtení	177
7.2.2 Opětovné načtení modulu s běhovou chybou	178
7.2.3 Opětovné načtení modulu s běhovou chybou	178
7.3 Získání odkazu na modul, atributy modulu.....	178
7.3.1 Získání odkazu na existující modul z názvu modulu.....	179
7.3.2 Načtení (import) modulu se zadaným názvem	179
<code>importlib.import_module(name, package=None)</code>	179
<code>importlib.invalidate_caches()</code>	179
7.3.3 Systémové atributy modulu	180
7.4 Koncepce balíčků a její důsledky.....	180
7.4.1 Fyzická a logická pozice modulu.....	180
7.4.2 Absolutní a relativní import.....	180
7.4.3 Modul nemusí znát svoji logickou lokaci.....	181
7.5 Virtuální prostředí Pythonu	181
7.5.1 Vytvoření virtuálního prostředí.....	182
7.5.2 Spuštění virtuálního prostředí.....	185
7.5.3 Přímé spuštění interpretu ve virtuálním prostředí.....	187
7.6 Kde hledat moduly.....	188
7.6.1 Přidání stromu složek při startu Pythonu – soubory <code>.pth</code>	188
Umístění souborů <code>.pth</code>	188
Obsah souborů <code>.pth</code>	189
Ukázka použití	190
7.7 Neinicializované balíčky – NS-balíčky	192
7.7.1 Teoretický úvod.....	192
NS-balíčky jsou pouze zobecněním pravidel pro kořenový balíček	193
NS-balíček smí mít pouze NS rodiče	193
7.7.2 Postup hledání modulu.....	193
7.7.3 Důsledky.....	194
7.7.4 Standardní balíček v NS-balíčku.....	194
7.7.5 Příklad.....	195
Modul <code>mA_importing</code>	195
Importované moduly	195
Spuštění	197
7.8 Shrnutí kapitoly a doprovodné soubory.....	198
7.8.1 Doprovodné soubory	199

Kapitola 8 Volatelné objekty a jejich vlastnosti	200
8.1 Co jsou volatelné objekty	200
8.1.1 Dosud probrané volatelné objekty	200
8.1.2 Jak udělat objekt volatelným	201
8.1.3 Jak definovat volatelný modul	202
8.2 ZNA a jeho rámce	203
8.3 Požadavky funkcionálního programování	205
8.4 Funkce jako plnohodnotné objekty, funkce vyššího řádu	206
8.4.1 Funkce jako návratová hodnota	206
8.4.2 Funkce jako parametr a argument	207
8.4.3 Použití modulu <code>operator</code> při předávání operátorů v parametru	208
8.5 Lambda-výrazy jako anonymní funkce	209
8.6 Uzávěry	209
8.6.1 Definice vnější a vnitřní funkce	210
8.6.2 Nutnost použití závěru	211
8.6.3 Prověra	211
8.6.4 Jak to celé funguje	211
8.7 Částečně vyhodnocené funkce	212
8.7.1 Použití lambda výrazů	213
8.7.2 Použití funkce <code>functools.partial()</code>	213
8.8 Shrnutí kapitoly a doprovodné soubory	214
8.8.1 Doprovodné soubory	214
Kapitola 9 Dekorátory	215
9.1 Co jsou dekorátory	215
9.1.1 Terminologie	216
9.1.2 „Operátor“ <code>@</code>	216
Dosavadní použití dekorátorů	217
9.1.3 Prefixové a postfixové použití dekorátoru	218
Dekorace externího objektu	218
Vytažení některých atributů do obálky	218
9.2 Tvorba jednoduchého dekorátoru	219
9.3 Dekorátor definovaný jako třída	220
9.4 Dekorátory s parametry	223
9.5 Současné použití více dekorátorů	223
9.6 Dekorátory třídy	224
9.6.1 Jedináček – singleton	224
9.7 Ukládání výsledků do vyrovnávací paměti	226
9.8 Shrnutí kapitoly a doprovodné soubory	228
9.8.1 Doprovodné soubory	228
Kapitola 10 Třídní metody	229
10.1 K čemu je dekorátor <code>classmethod</code>	229
10.2 Tovární metody	231
10.3 Ovlivnění tvorby dceřiných tříd	232
Definice třídy <code>C04a</code>	233
Definice potomků	235
10.3.1 Zákaz definice potomků dané třídy	235
10.4 Jedináček definovaný pomocí <code>__new__()</code>	235
10.5 Komplexnější použití třídních metod	236
10.5.2 Návrhový vzor Multiton	236
10.5.3 Definice společné rodičovské třídy	237
10.5.4 Prověra	239
10.6 Shrnutí kapitoly a doprovodné soubory	240
10.6.1 Doprovodné soubory	240

Kapitola 11	Iterátory a generátory	241
11.1	Iterovatelné objekty – iterables	241
11.2	Iterátory	242
	<code>__iter__(self)</code>	242
	<code>__next__(self)</code>	242
11.2.1	Příklad	243
11.3	Generátorový výraz	244
11.4	Nekorektní použití generátorového výrazu	245
11.5	Generátorová funkce a generátorový iterátor	245
11.6	Definice generátorové funkce jako zdroje	247
11.7	Použití více příkazů <code>yield</code> v těle funkce	248
11.8	Vestavěné funkce související s iteracemi	249
	<code>iter(object[, sentinel])</code>	249
	<code>map(function, iterable, ...)</code>	249
	<code>next(iterator[, default])</code>	250
11.9	Jak to celé funguje	250
11.10	Operátor <code>yield</code> – <code>yield</code> jako výraz	250
	Analogie – AlzaBox	251
11.10.1	Demonstrace ruční aktivace generátoru s výrazem <code>yield</code>	251
	Popis funkce <code>counter</code>	252
	Použití funkce <code>counter</code>	253
11.10.2	Demonstrační AHA-příklad použití výrazu <code>yield</code>	254
	Generátorová funkce <code>ord_gen()</code>	254
	Funkce <code>orders()</code>	255
11.10.3	Metoda <code>send()</code> nemusí zadávat jen čísla	257
11.11	Výraz <code>yield from</code>	257
11.11.1	Trochu složitější příklad	258
11.12	Shrnutí kapitoly a doprovodné soubory	259
11.12.1	Doprovodné soubory	259
Kapitola 12	Výčtové typy	260
12.1	Výčtové typy nejsou součástí definice jazyka	260
12.2	Další možnosti výčtových typů	261
12.2.1	Definice „obyčejných“ atributů	261
12.2.2	Přezdívkový hodnot	262
12.2.3	Automatické přiřazení obalovaných hodnot	264
12.2.4	Poloautomatické přiřazení obalovaných hodnot	265
	Poznámka k vývoji Pythonu	266
12.3	Zveřejnění prvků – dekorátor <code>global_enum</code>	266
12.4	Výměna systémových metod	268
12.5	Útroby výčtového typu	270
12.5.1	Výčtové typy jako programový incest	270
	Shrnutí	271
12.5.2	Interní atributy výčtových typů a jejich instancí	271
12.5.3	Užitečné interní metody	273
	<code>_generate_next_value(name, start, count, last_values)</code>	273
	<code>_new_member(cls, *args)</code>	275
	<code>_missing(cls, value)</code>	275
	<code>_new__(cls, *args)</code>	275
12.5.4	Využití definice vlastního alokátoru <code>_new__()</code>	275
	Definice alokátoru – metody <code>_new__()</code>	277
	Definice zbylých metod	277
	Vytvoření objektu třídy D4	278
	Prověrka funkcionality	278

Proč nestačilo definovat <code>__init__()</code>	279
Definice třídy D4x	279
Vytvoření objektu třídy D4x	279
Porovnání atributů tříd D4 a D4x	279
Závěr	281
12.6 Výčtové typy s více předky	281
12.7 Odvozené výčtové typy	281
Co to jsou mixiny a k čemu jsou dobré	282
12.7.1 Prázdné výčtové typy	283
12.7.2 <code>ReprEnum</code>	284
12.7.3 <code>IntEnum</code>	284
12.7.4 <code>StrEnum</code>	285
12.7.5 <code>Flag</code>	286
12.7.6 Reakce na hodnoty mimo rozsah – třída <code>FlagBoundary</code>	287
Barevné systémy	288
<code>STRICT</code>	288
<code>CONFORM</code>	289
<code>EJECT</code>	289
<code>KEEP</code>	290
12.7.7 <code>IntFlag</code>	290
12.8 Shrnutí kapitoly a doprovodné soubory	291
12.8.1 Doprovodné soubory	291
Kapitola 13 Datové třídy	292
13.1 Představení	292
13.2 Definice	292
<code>dataclass(*, init=True, repr=True, eq=True, order=False,</code> <code>unsafe_hash=False, frozen=False, match_args=True, kw_only=False,</code> <code>slots=False, weakref_slot=False)</code>	293
13.2.1 Omezení implicitních počátečních hodnot	295
Pořadí	295
Proměnnost	295
13.2.2 Užitečné datové atributy	296
<code>KW_ONLY</code>	296
<code>MISSING</code>	297
13.3 Funkce <code>field()</code>	297
<code>field(*, default=MISSING, default_factory=MISSING, init=True,</code> <code>repr=True, hash=None, compare=True, metadata=None, kw_only=MISSING)</code>	297
Příklady použití funkce <code>field()</code> s argumentem <code>default_factory</code>	298
13.4 Vylepšujeme vytváření instancí	300
13.4.1 Třídní datové atributy	300
Vliv třídního atributu na implicitní hodnotu	300
Zdůraznění třídnosti atributu v anotaci	301
13.4.2 Pole v hierarchii dědění	301
13.4.3 Přerovnávání zděděných polí při povinném pojmenování	302
13.4.4 Metoda <code>__post_init__()</code>	303
Doplnění volání initoru předka	303
13.4.5 Inicializační proměnné – anotace <code>InitVar</code>	304
13.5 Další užitečné funkce	305
<code>asdict(obj, *, dict_factory=dict)</code>	306
<code>astuple(obj, *, tuple_factory=tuple)</code>	306
<code>replace(obj, /, **changes)</code>	306
13.6 Shrnutí kapitoly a doprovodné soubory	307
13.6.1 Doprovodné soubory	307

Kapitola 14	Anotace a koncept typového systému	308
14.1	Trocha historie	308
14.1.1	Výhody používání typových anotací	309
14.2	Statické a dynamické testování	310
	Dynamické testování	310
	Statické testování	310
14.2.1	Nejpoužívanější statické analyzátoru a testery	311
	MyPy	311
	Pyright	311
	Pylance	311
	MonkeyType / Pyannotate	312
	Shrnutí	312
14.2.2	Nejpoužívanější knihovny pro tvorbu dynamických testů	312
14.3	Zápis anotace	312
14.4	Deklarace běžných typů	314
14.4.1	Jednoduché typy	314
14.4.2	Kontejnery	314
14.4.3	Volitelné objekty	315
14.4.4	Zabudované datové typy a jejich přezdívky („aliasy“)	316
14.5	Atribut <code>__annotations__</code>	317
14.5.1	Geneze odloženého vyhodnocování anotací	318
	Okamžité vyhodnocování	318
	Stringová sebeobrana	318
	Postponed evaluation	318
	Deferred evaluation	318
14.5.2	Starší řešení – postponed evaluation	319
	Modul <code>__future__</code>	320
14.5.3	Současné implicitní řešení (deferred evaluation)	322
	Přepnutí do režimu postponed	322
14.6	Modul <code>typing</code>	324
14.6.1	Starší definice přezdívek typů – funkce <code>TypeAlias</code>	324
14.6.2	Moderní definice přezdívek typů – příkaz <code>type</code> (Python 3.12+)	325
	Příkaz <code>type</code> a generické typy	325
14.6.3	Sjednocení datových typů – třída <code>Union</code>	326
	Datový typ <code>Optional[x]</code>	327
14.6.4	Třída <code>TypeVar</code> a její použití	327
	Konstruktor	327
	<code>TypeVar(name, *constraints, bound=None, covariant=False, contravariant=False)</code>	328
	Použití	328
14.6.5	Zavedení nového datového typu – třída <code>NewType</code>	329
14.6.6	Několik užitečných anotací	331
	<code>final</code>	331
	<code>override</code>	331
	<code>LiteralString</code>	332
	<code>Never</code> <code>NoReturn</code>	332
	<code>Self</code>	332
14.7	Postupné typování (gradual typing)	332
14.7.1	Datový typ <code>Any</code>	333
14.7.2	Funkce <code>cast()</code>	334
14.8	Zdánlivé přetěžování funkcí a metod	334
14.9	Shrnutí kapitoly a doprovodné soubory	337
14.9.1	Doprovodné soubory	337
Kapitola 15	Generické datové typy a generické funkce	338
15.1	Představení	338

15.1.1 Používání generických datových typů.....	339
15.2 Definice vlastních generických datových typů.....	339
15.2.1 Starší podoba definice generického typu (Python 3.5+)	340
15.2.2 Současná podoba definice generického typu (Python 3.12+)	341
15.2.3 Definice generického typu s více typovými proměnnými	342
15.2.4 Generické přezdívky typů – příkaz <code>type[T]</code>	342
Pojmenování tvaru místo rozepisování detailů	343
Přehlednost API a snadná změna na jednom místě.....	343
Neplést s anotačním zápisem <code>type[T]</code>	344
15.3 Kladení požadavků na typové proměnné	344
15.3.1 Omezení typových parametrů na potomky	344
15.3.2 Výčet možných typů	345
15.3.3 Výchozí hodnoty typových parametrů (Python 3.13+).....	347
15.3.4 Další možnosti dosažitelné pomocí <code>TypeVar</code>	347
Kovariance	348
Kontravariance	348
Odvození variance	349
15.4 Generické funkce a metody	349
15.5 Pomocné funkce pro výstižnější anotace	349
<code>Final</code>	350
<code>Literal</code>	350
<code>Optional</code>	351
<code>TypeGuard</code>	351
15.6 Funkce <code>typing.Annotated</code>.....	352
15.6.1 Motivace k zavedení funkce <code>Annotated</code>	353
15.6.2 K čemu se třída <code>Annotated</code> používá.....	354
15.6.3 Příklad zadání metadat pro datovou třídu	354
15.6.4 Demo běhové verifikace objektu pomocí metadat z <code>Annotated</code>	355
15.7 Shrnutí kapitoly a doprovodné soubory.....	357
15.7.1 Doprovodné soubory.....	357
Kapitola 16 Výjimky a varování.....	358
16.1 Anatomie výjimky	358
16.1.1 Proč <code>BaseException</code>	358
16.1.2 Důležité atributy	359
<code>args</code>	359
<code>__cause__</code>	359
<code>__context__</code>	359
<code>__notes__</code>	359
<code>__suppress_context__</code>	359
<code>__traceback__</code>	359
Zřetězený seznam	360
16.1.3 Důležité metody	360
<code>__str__()</code>	361
<code>__repr__()</code>	361
<code>add_note(note:str) -> None</code>	361
<code>with_traceback(tb)</code>	361
16.2 Doplnění výjimky o poznámku	361
16.2.1 Kód chyby.....	362
16.3 Příkaz <code>raise...from</code> a kontext výjimky	363
16.3.1 Příkaz <code>raise ... from None</code> a vyhození ošetřované výjimky	365
16.4 Definice vlastní reakce na neošetřené výjimky	366
16.4.1 Kdy definovat vlastní <code>excepthook</code>	367
16.4.2 Kdy vlastní <code>excepthook</code> nedefinovat	367
16.4.3 Závěrečná doporučení a upozornění.....	367

Uložení a volání původního hooku	367
Robustnost samotného excepthook	368
Zabalení hlavní funkce programu	368
Frameworky	368
Služby pro reportování chyb	368
16.5 Skupinové výjimky a větve except*	368
16.5.1 Zavedené třídy a větve except*	369
16.5.2 Definice skupinové výjimky	369
16.5.3 Atributy a metody třídy BaseExceptionGroup	371
message	371
exceptions	371
subgroup(condition) -> typing.Self	371
split(condition) -> tuple[typing.Self, typing.Self]	371
derive(excs) -> typing.Self	371
__traceback__	373
Poznámka o konstruktorech a metodě __new__()	373
16.5.4 Příklad: použití v definici funkce řešící komplexní problém	373
16.5.5 Závěrečná doporučení	373
16.6 Test, zda se právě neošetřuje výjimka	374
Funkce sys.exception()	376
Závěrečné doporučení	376
16.7 Získání informací o volajících	376
16.7.1 ZNA a jeho rámce	376
Objekty typu types.FrameType	376
Objekty typu types.CodeType	377
16.7.2 Třída traceback	377
16.7.3 Využití modulu inspect – nahlédnutí pod kapotu Pythonu	378
Funkce stack() a instance třídy inspect.FrameInfo	378
Příklad využití modulu inspect při tvorbě dekorátoru:	379
Prověra vytvořeného dekorátoru	382
16.8 Shrnutí kapitoly a doprovodné soubory	383
16.8.1 Doprovodné soubory	384

Část C Pokročilejší objektové konstrukce

385

Kapitola 17 Přetěžování operátorů	386
17.1 Přetěžování metod versus přetěžování operátorů	386
17.2 Základy	387
17.2.1 Neimplementované operace – NotImplemented	387
17.3 Základní sada	388
17.3.1 Reprezentace objektu a jeho logická hodnota	388
__bool__(self)	388
__repr__(self), __str__(self)	388
__format__(self, format_spec)	388
17.3.2 Objekty reprezentující hodnotu (ORV) a objekty reprezentující entitu (ORE)	389
17.3.3 Neměnnost ORV objektů	390
17.3.4 Porovnávání objektů	390
__lt__(self, other) __le__(self, other) __eq__(self, other)	391
__ne__(self, other) __gt__(self, other) __ge__(self, other)	391
Není třeba definovat všechny	391
Demonstrační příklad	391
Interpret chápe metody obecněji	393
Obecnější mohou být i návratové hodnoty	393
17.3.5 Hešování objektů	394
__hash__()	394

17.3.6 Finalizace objektu	394
__del__()	394
Destruktor versus finalizér	395
17.4 Operátory + - * @ / // % ** << >> & ^ a emulace numerických typů	395
17.4.1 Binární operátory	396
Pravostranné (reflexivní) operátory	396
17.4.2 Unární operátory	397
17.4.3 Zbylé emulační funkce	397
17.4.4 Konverzní funkce	398
17.4.5 Souhrnný příklad: třída Zlomek	398
Přezdívky typů	403
Initor	403
Reprezentační metody	403
Konverzní metody	403
Unární operátory	404
Porovnávací operátory	404
Binární aritmetické operátory	404
Rozšířené přiřazení	404
17.4.6 Prověra správnosti definice třídy Zlomek	405
17.5 Operátor [], emulace indexovatelných kontejnerů	406
__contains__(self, item)	406
__delitem__(self, key)	407
__getitem__(self, key)	407
__iter__(self)	407
__len__(self)	407
__missing__(self, key)	407
__reversed__(self)	407
__setitem__(self, key, value)	407
17.5.1 Příklad	407
Prověra třídy PosloupnostMocnin	408
17.6 Operátor () – volatelné objekty	409
17.6.1 Předehra	410
17.6.2 Jednoduchý příklad	410
17.6.3 Něco trochu složitějšího	410
17.7 Další užitečné dunderly	412
17.7.1 Metody pro kopírování objektů	413
__copy__(self)	413
__deepcopy__(self, memo)	413
17.7.2 Metoda pro konverzi na index	413
__index__(self)	413
17.7.3 Metody pro zaokrouhlování	413
__round__(self, ndigits=None)	413
__trunc__(self)	414
__floor__(self)	414
__ceil__(self)	414
Závěr	414
17.8 Shrnutí kapitoly a doprovodné soubory	414
17.8.1 Doprovodné soubory	415
Kapitola 18 Ovlivnění přístupu k atributům	416
18.1 Předehra	416
18.2 Co jsou deskriptory	417
__get__(self, instance, owner=None)	417
__set__(self, instance, value)	417
__delete__(self, instance)	417
__set_name__(self, owner, name)	417

18.2.1	Demonstrační příklad.....	418
	Použití definovaných tříd	419
18.3	Dělení deskriptorů	419
18.3.1	Datové deskriptory	420
18.3.2	Nedatové deskriptory	420
18.4	Definice ekvivalentu třídy property.....	420
18.5	Deskriptor spravující hodnotu.....	423
18.5.1	Test vytvořeného deskriptoru.....	424
18.6	Pořadí vyhodnocování	426
	Proč XXX a ne None	426
	1. Datový deskriptor.....	427
	2. Atribut instance.....	427
	3. Nedatový deskriptor.....	427
	4. Řadový atribut třídy	428
	5. Chyba	428
18.7	Příklad: deskriptor pro odložené vyhodnocení	428
18.8	Jak definovat vlastnosti třídy	430
18.8.1	Kudy cesta nevede.....	430
18.8.2	Definice třídy classproperty	431
18.8.3	Demonstrace užití třídy classproperty	431
	Robustnost	433
	Překvapivé chování del.....	433
18.9	Operátor . (tečka) – přístup k atributům	434
	__getattr__(self, name)	434
	__getattr__(self, name)	434
	__setattr__(self, name, value)	434
	__delattr__(self, name)	435
	__dir__(self)	435
18.9.1	AHA-příklad – definice	435
18.9.2	AHA-příklad – prověrka	435
	Aktivita IDLE	436
	IDLE versus jiná prostředí	438
	Další příkazy.....	439
18.9.3	Lepší řešení.....	439
18.10	Jaký způsob správy přístupu k atributu zvolit.....	440
18.11	Ovlivnění přístupu k atributům modulu.....	441
18.12	Shrnutí kapitoly a doprovodné soubory.....	444
18.12.1	Doprovodné soubory	444
Kapitola 19	Správa paměti a pokročilé techniky práce s objekty.....	445
19.1	Co už víme z prvního dílu	445
	Vše je objekt	445
	Analogie: proměnná je telefon a odkaz je telefonní číslo	446
	Správa paměti	446
19.1.1	Proč řešit paměť a životní cyklus objektů	446
19.2	Základy správy paměti	446
19.2.1	Počítání odkazů	447
	Analogie s telefonními čísly	447
19.2.2	Architektura správy paměti v Pythonu	447
19.2.3	Správce paměti (Memory Manager).....	447
19.2.4	Uklízeč (Garbage Collector)	447
19.2.5	Spolupráce uklízeče a správce paměti	448
19.3	Slots	448
19.3.1	Výhody a nevýhody slotů	449
19.3.2	Kdy __slots__ použít a kdy je nepoužívat	450
19.4	Slabé odkazy	450

19.4.1	Problém s nechtěným udržováním objektů	451
19.4.2	Řešení: weakref – odkaz bez závazků	452
19.5	Propojení slotů a slabých odkazů	454
19.6	Shrnutí kapitoly a doprovodné soubory	455
19.6.1	Doprovodné soubory	455
Kapitola 20	Ovlivnění tvorby tříd, metatřídy	456
20.1	Přehled k výkladu metatříd	456
20.2	Co jsou to metatřídy	457
20.3	Postup při vytváření třídy	458
20.4	Různé definice metatříd	460
20.4.1	Vzorový kód pro demonstraci funkce metatřídy	460
20.4.2	Jednoduchý úvod s metatřídou definovanou jako funkce	462
	Import modulu <code>modules.m20b_MetaSimple</code>	463
20.4.3	Kompletní příklad s metatřídou definovanou jako funkce	464
	Import modulu <code>modules.m20c_MetaFun</code>	465
20.4.4	Metatřída definovaná jako třída	466
	Import modulu <code>modules.m20d_MetaCls</code>	467
20.4.5	Metatřída definovaná jako objekt	469
	Import modulu <code>modules.m20e_MetaObj</code>	470
20.4.6	Závěr	471
20.5	Jedináček definovaný prostřednictvím metatřídy	472
20.6	Potřeba postfixové dekorace třídy	473
20.6.1	Import modulu	475
20.7	Shrnutí kapitoly a doprovodné soubory	476
20.7.1	Doprovodné soubory	476

Část D Programování souběžných úloh

477

Kapitola 21	Principy souběžného programování	478
21.1	Souběžné provádění více úloh	478
21.1.1	Souběžnost versus paralelismus	479
	Souběžnost (concurrency)	479
	Paralelismus (parallelism)	479
	Rozdíly	479
21.2	Plánovač, vlákna a strategie plánování	479
21.2.1	Kooperativní plánování	480
21.2.2	Preemptivní plánování	480
21.3	Procesy – vlákna – koprogramy – subinterprety	481
21.3.1	Procesy	481
21.3.2	Vlákna	482
21.3.3	Koprogramy a korutiny	482
	Historická vsuvka o vývoji terminologie	483
21.3.4	Subinterprety (novinka Pythonu 3.14)	483
21.4	Kategorizace řešených problémů	484
21.4.1	Úlohy vytěžující CPU – CPU-bound	484
21.4.2	Úlohy vytěžující vstup a/nebo výstup – IO-bound	485
21.5	Komunikace mezi programovými jednotkami	485
21.5.1	Předávání zpráv	485
21.5.2	Sdílená paměť	485
21.5.3	Kritická sekce a souběh	486
	Hazardy	487
21.5.4	Hybridní řešení	487
21.6	Synchronizace	488
21.6.1	Explicitní zámky	488
	Zámek pro čtenáře a zapisovatele	488

21.6.2	Monitory a implicitní zamykání	489
21.6.3	Rendezvous	489
21.6.4	Co dál?	489
21.7	Stavový diagram chování vláken	490
21.7.1	Analogie	491
21.7.2	Skutečnost je maličko komplexnější	492
21.8	Přístup jazyka Python	492
21.8.1	Další vývoj GIL	493
	Projekt nogil	493
	Subinterprety	493
21.9	Srovnání s jazyky Java a C#	494
21.10	Otevření dveří pro knihovny v dalších jazycích	494
21.10.1	Projekt HPy: nové, univerzálnější API	495
21.11	Shrnutí kapitoly a doprovodné soubory	495
21.11.1	Doprovodné soubory	496
Kapitola 22	Vlákna a modul threading	497
22.1	Přehled	497
22.2	Znovu Karel	498
22.2.1	Modul util.karelcz_lib	498
22.2.2	Funkce multithread()	498
	Proč to děláme tak složitě	499
22.3	Třída Thread	499
22.3.1	Konstruktor instancí třídy Thread	500
	Thread(group=None, target=None, name=None, args=(), kwargs={}, *, daemon=None, context=None)	500
22.3.2	Důležité atributy a metody třídy Thread	500
	start()	500
	run()	501
	join(timeout=None)	501
	name	501
	is_alive()	501
	daemon	501
22.4	Nezávislá vlákna	502
22.4.1	Vytvoření světa se čtyřmi dvorky	502
22.4.2	Příprava na úkol	503
22.4.3	Zabalení a spuštění úkolu současně pracujících robotů	504
22.5	Nezávislé čekání do doběh vlákna	505
22.5.1	Modifikované zadání	505
22.5.2	Zřetězení činností vláken	506
	Závěr	508
22.6	Nesynchronizované sdílení prostředků	508
22.6.1	Scénář se sdíleným zdrojem	509
22.6.2	Výchozí funkce pro dodavatele a odběratele	509
	Dodavatel	509
	Odběratel	509
22.6.3	Propojení nesynchronizovaných činností	510
22.6.4	Problémy nesynchronizovaných činností	512
22.7	Kritické sekce a zámky	513
22.7.1	Třída threading.Lock	514
	acquire(blocking=True, timeout=-1)	514
	release()	514
	Příklad	515
22.7.2	Skrytý pozůstatek démona	516
22.7.3	Správce kontextu a příkaz with	517
22.7.4	Úprava architektury	517

22.8 Shrnutí kapitoly a doprovodné soubory.....	518
22.8.1 Doprovodné soubory.....	519
Kapitola 23 Podprocesy a modul multiprocessing.....	520
23.1 Základní kameny modulu multiprocessing	520
23.1.1 Procesy a podprocesy	520
23.1.2 Oddělená paměť	521
23.1.3 Komunikace mezi procesy.....	521
23.1.4 Serializace	522
23.1.5 Třída multiprocessing.Process.....	522
23.1.6 Ochrana hlavního modulu: if <code>__name__ == "__main__":</code>	522
23.2 Klíčové objekty	523
23.2.1 Třída multiprocessing.Process.....	523
<code>Process(group=None, target=None, name=None, args=(), kwargs={}, *,</code> <code>daemon=None)</code>	523
<code>run()</code>	523
<code>start()</code>	523
<code>join(timeout=None)</code>	523
<code>name</code>	524
<code>is_alive()</code>	524
<code>daemon</code>	524
<code>pid</code>	524
<code>terminate()</code>	524
<code>close()</code>	524
23.2.2 Třída multiprocessing.Queue.....	524
<code>Queue(maxsize=)</code>	525
<code>qsize()</code>	525
<code>empty()</code>	525
<code>put(obj, block=True, timeout=None)</code>	525
<code>put_nowait(obj)</code>	525
<code>get(block=True, timeout=None)</code>	525
<code>get_nowait()</code>	525
<code>close()</code>	525
<code>join_thread()</code>	525
<code>cancel_join_thread()</code>	525
23.2.3 Tovární funkce multiprocessing.Pipe()	526
23.2.4 Třída multiprocessing.connection.Connection.....	526
<code>send(obj)</code>	526
<code>recv()</code>	526
<code>close()</code>	526
23.2.5 Globální atributy modulu multiprocessing.....	526
<code>active_children() -> list[Process]</code>	526
<code>cpu_count() -> int</code>	526
<code>current_process() -> Process</code>	526
<code>parent_process() -> Process</code>	527
23.2.6 Mrtvý proces je ještě potřeba „pohřbit“	527
23.3 Demonstrační příklad	527
23.3.1 Demonstrační programy budou moduly	527
23.3.2 Nefunkční přístup ke sdílení paměti	527
Kontrolní tisky	529
Hlavní proces	529
Nešlo by to jednodušeji?	530
Podproces – funkce <code>worker()</code>	530
Spouštěcí blok	530
Spuštění programu	530

Závěr.....	532
23.3.3 Funkční řešení pomocí Queue.....	532
Tělo modulu.....	532
Hlavní proces – funkce <code>main()</code>	533
Podproces – funkce <code>worker()</code>	534
Spuštění	534
Možná překvapení	534
23.4 Knihovna <code>KarelCz7475</code> a její GUI.....	535
23.4.1 Návrhový vzor Posluchač.....	535
23.4.2 Základní architektura knihovny <code>KarelCz7475</code>	536
23.4.3 Jak implementovat posluchače v odděleném procesu	536
23.4.4 Návrh zástupce a jeho partnera.....	537
23.4.5 Demonstrace funkce zástupce.....	537
23.4.6 Návrh komunikace	537
23.4.7 GUI server	539
23.4.8 Testovací aplikace.....	541
Spuštění	542
23.5 Vícevláknová aplikace se vzdáleným GUI.....	543
23.6 Shrnutí kapitoly a doprovodné soubory.....	545
23.6.1 Doprovodné soubory	545
Kapitola 24 Korutiny a modul <code>asyncio</code>	546
24.1 Knihovna/modul <code>asyncio</code>	546
24.2 Koprogramy, korutiny a očekávatelné objekty	547
24.3 Definice a použití korutin.....	548
24.3.1 Vytvoření a spuštění korutiny	549
24.3.2 Čekání na jinou korutinu	549
24.3.3 Využití objektů typu <code>Task</code>	551
24.3.4 Správa skupiny úloh – <code>gather()</code> a <code>TaskGroup</code>	552
24.3.5 Třída <code>TaskGroup</code> a příkaz <code>async with</code>	553
Příkaz <code>async with</code>	553
Třída <code>asyncio.TaskGroup</code>	553
24.3.6 Nejčastější chyba – zapomenutý <code>await</code>	554
Zlaté pravidlo <code>asyncio</code>	555
24.4 Úprava knihovny pro asynchronní provoz.....	555
Podrobněji o <code>asyncio</code> a jeho korutinách.....	556
Role <code>await</code> a smyčky událostí	556
Proč to nelze volat synchronně?	556
24.4.1 Postup doplnění podpory <code>asyncio</code>	557
24.4.2 Aktivační modul <code>util.karelCz_async</code>	558
Zákaz řetězení asynchronních volání	559
Synchronní metody v asynchronním světě	560
24.4.3 Prověrka funkce	560
24.5 Asynchronní cyklus – příkaz <code>async for</code>	561
24.6 Asynchronní zámky a synchronizace.....	562
24.6.1 Třída <code>asyncio.Lock</code> a příkaz <code>async with</code>	563
24.7 Shrnutí kapitoly a doprovodné soubory.....	565
24.7.1 Doprovodné soubory	566
Kapitola 25 Subinterprety a modul <code>concurrent.interpreters</code>	567
25.1 Předehra	567
25.1.1 Dva režimy běhu subinterpretů.....	568
Režim A – lokálně v témže vlákne	568
Režim B – v samostatném vlákne	568
Analogie.....	568
25.2 Základní operace: vytvoření a ukončení	568

25.2.1 Pomocná metoda <code>counter()</code>	569
25.2.2 Demonstrační program	570
25.2.3 Společná konzole (<code>stdin</code> , <code>stdout</code> , <code>stderr</code>)	571
Výstup	572
Vstup	572
Proč jsme to nevyužívali v procesech	572
25.3 Předávání dat – opět Karel	572
25.3.1 Architektura řešení	573
Místo datovodu použijeme frontu	574
Komunikace prostřednictvím zástupce	574
Modul <code>IO</code>	574
25.3.2 Zavaděč knihovny <code>KarelCz7475</code> pro subinterprety	574
Hlavní funkce modulu	575
Zaváděcí funkce	575
Tvorba náhradního modulu <code>karelCz.IO</code>	577
Začlenění vytvořeného modulu	577
25.3.3 Lokální zástupce	577
Funkce v roli zavaděče	579
Spuštění metody	580
25.3.4 Vzdálený zástupce – dispečer	580
25.3.5 Koordinátor	582
25.3.6 Jednoduchý test	582
Nastavení podrobnosti ladících informací	582
Import funkce <code>run_in_si()</code>	585
Import funkce <code>simple_demo()</code>	585
Příkaz <code>run_in_si(simple_demo)</code>	585
Činnost funkce <code>run_callable_in_si()</code>	586
25.4 Subinterpret a vícevláknový režim	587
25.4.1 Třída <code>threading.Event</code>	587
<code>is_set()</code>	587
<code>set()</code>	587
<code>clear()</code>	587
<code>wait(timeout:float=None)</code>	587
25.4.2 Úprava testovacích funkcí – modul <code>m25f_multithread</code>	588
25.5 Shrnutí kapitoly a doprovodné soubory	590
25.5.1 Doprovodné soubory	590

Literatura	591
Rejstřík	593