Část F
Přílohy

V této části naleznete čtyři přílohy: první vám vysvětlí, jak lze ve *Windows* používat substituované disky, druhá vám představí syntaktické diagramy, třetí vás seznámí se základními konvencemi pro psaní programů v *Pythonu* a odkáže vás na dokumenty, v nichž je dané téma rozebráno podrobněji, a čtvrtá poskytne stručný přehled základních inovací v několika posledních subverzích jazyka.

Příloha A
Konfigurace ve Windows

A.1 Definice substituovaných disků

V operačním systému *Windows* můžete používat 26 logických disků – pro každé písmeno abecedy jeden. Většina uživatelů však používá pouze zlomek tohoto počtu. *Windows* umožňují použít volná písmena pro tzv. **substituované disky**, což jsou složky, které se rozhodnete vydávat za logický disk. Protože o této možnosti většina uživatelů neví, a přitom je to funkce velice užitečná, ukážu vám, jak ji můžete využít. Substituované disky se definují pomocí příkazu:

```
SUBST název_disku substituovaná_složka
```

Nejjednodušší způsob, jak definovat ve *Windows* např. substituovaný disk `P:`, je vložit do složky, kterou budete chtít substituovat jako disk `P:`, dávkový soubor s příkazem k substituci. (Písmeno `P` se pro *Python* hodí nejlépe, ale můžete si vybrat jakékoliv jiné, které je na vašem počítači volné.)

Pokud jste ještě nepracovali s dávkovými soubory, tak vězte, že to jsou obyčejné textové soubory, do nichž zapisujete příkazy pro operační systém. Jejich název může být libovolný, ale musí mít příponu bat (zkratka ze slova batch – dávka). V dávkovém souboru budou následující příkazy (na velikosti písmen nezáleží):

```
SUBST P: /d
SUBST P: .
```

První příkaz má za úkol zrušit případnou doposud nastavenou substituci disku `P:` (není-li v daném okamžiku označený disk substituován, systém vypíše chybovou zprávu, ale jinak se nic nestane), druhý příkaz pak substituuje aktuální složku jako disk `P:`.

Soubor umístíte do složky, z níž budete chtít udělat substituovaný disk. Kdykoliv pak tento dávkový soubor spustíte, dávka substituuje složku, v níž je umístěna, jako příslušný disk. Dávka se přitom spouští obdobně jako aplikace – např. poklepáním na ikonu jejího souboru v *Průzkumníku*.

Kdykoliv od této chvíle budete pracovat s diskem `P:`, budete ve skutečnosti pracovat s obsahem substituované složky. A naopak: cokoliv uděláte s obsahem substituované složky, uděláte zároveň s obsahem disku `P:`.

Budete-li chtít mít danou substituci nastavenu trvale, můžete umístit zástupce dávkového souboru do položky Po spuštění ve startovní nabídce. Protože je v každé verzi operačního systému jinde, bude nejlepší, když ji ve startovní nabídce najdete, klepnete na ni pravým tlačítkem a v následně otevřené místní nabídce zadáte **Otevřít**. Tím otevřete okno průzkumníka s touto složkou. Pak v druhém okně průzkumníka otevřete složku s příslušným dávkovým souborem, uchopíte jeho ikonu PRAVÝM tlačítkem myši, přesunete ji do složky nabídky a pustíte. Otevře se místní nabídka (ta se otevře, pouze pokud přesouváte soubor pravým tlačítkem myši), ve které zadáte, že zde chcete vytvořit zástupce, a tím celý proces končí.

Abyste mohli složku substituovat jako nějaký disk, nesmí váš operační systém používat disk označený tímto písmenem. Písmeno může být použito nejvýše pro jiný substituovaný disk, protože tuto substituci můžete před nastavením nové substituce zrušit (pro tento případ je v dávkovém souboru první příkaz s parametrem `/d` – delete).

Používáte-li operační systém *Windows*, můžete urychlit budoucí vyhledání složky s projekty právě tím, že pro ni zřídíte zvláštní substituovaný disk. Kdykoliv se pak obrátíte na příslušný disk, obrátíte se ve skutečnosti k příslušné složce. Pomocí substituce si tak můžete zkrátit cestu k často používaným složkám.

A.2 Nastavování zástupce spouštějícího IDLE

Používáte-li *Windows* a nemáte-li zkušenosti s nastavováním zástupců, doporučuji využít mého zástupce a pouze mu upravit některá nastavení. Postupujte následovně:

1. Otevřete ve svém oblíbeném správci souborů (budu předpokládat, že jím je *Průzkumník* nebo *Total Commander*) složku s doprovodnými programy `68_PYT`.
2. Klepněte na soubor `!IDLE_Python_39.lnk` (používáte-li *Průzkumník*, tak ten příponu zástupců, tj. `lnk`, nezobrazuje).
3. Stiskněte ALT+ENTER. Otevře se okno z obrázku A.1, v němž můžete nastavit parametry zástupce.
4. V poli **Cíl** je zadáno:

```
C:\_PGM\Python\Python39\pythonw.exe
C:\_PGM\Python\Python39\Lib\idlelib\idle.pyw
```

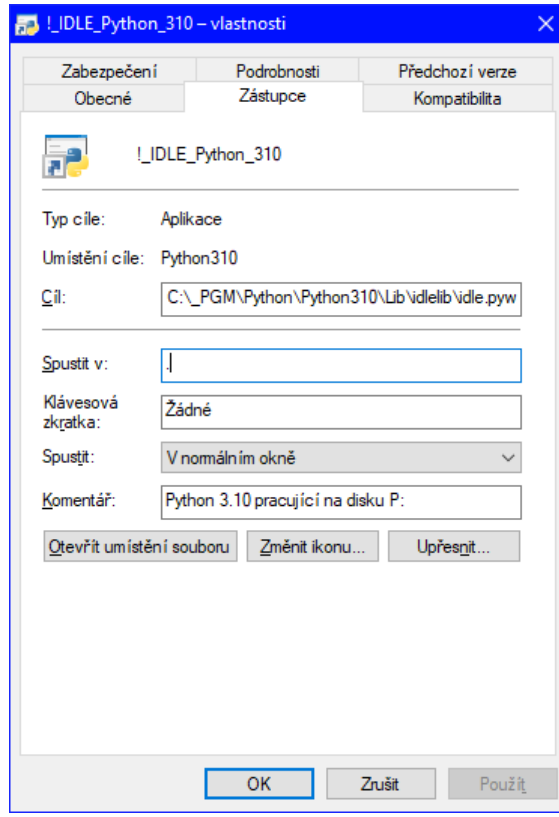
   První část končící `pythonw.exe` je úplná cesta k programu spouštějícímu *Python*, který nebude otevírat okno příkazového řádku, protože očekává, že spouštěný skript používá GUI.

   Tato část je nepovinná. Máte-li korektně instalovanou pouze jednu verzi *Pythonu*, nemusíte ji vůbec zadávat, protože *Windows* poznají, co mají spustit, podle přípony spouštěného skriptu. Já ji zadávat musím, protože mám v době

psaní knihy instalovanou vedle stabilního *Pythonu 3.8.3* také beta verzi *Pythonu 3.9*, kterou v této učebnici používám, a potřebuji si být jistý, která verze daný skript spouští.

Potřebujete-li proto stejně jako já také zadávat spouštěný program, musíte zde zadat úplnou cestu k tomuto programu na vašem počítači.

5. Druhá část příkazu je již povinná a zadává úplnou cestu ke spouštěnému skriptu, tj. k souboru `idle.pyw`. Vy budete mít spouštěný skript nejspíš jinde, tak zde musíte zadanou cestu příslušně upravit.
6. V poli **Spustit v** je zadána cesta ke složce se zdrojovými soubory doprovodných skriptů. Tu si také upravte podle svého počítače.
7. Když máte vše korektně upraveno, uložte upraveného zástupce stiskem **OK**, a od této chvíle můžete IDLE spouštět poklepáním na daného zástupce.



Obrázek A.1:
Okno zástupce spouštějícího IDLE

Příloha B
Syntaktické diagramy

Syntaktická pravidla zápisu složitějších konstrukcí jazyka jsou definována prostřednictvím syntaktických diagramů, které poskytují nejnázornější způsob jejich popisu. Tyto diagramy se v sedmdesátých letech minulého století používaly poměrně běžně, ale vzhledem k pracnosti jejich tvorby se následně z učebnic programování poněkud vytratily a v dnešní době se s nimi setkáte spíše výjimečně.

To ale nic nemění na skutečnosti, že jsou stále nejnázornějším popisem, prostřednictvím nějž si může začátečník ověřit, jak přesně vypadá syntaxe použité konstrukce a kde pravděpodobně udělal ve svém programu chybu. Proto je také ve svých učebnicích používám.

Syntaktické diagramy jsou grafickým ekvivalentem zápisu v EBNF[58] používaném v převážné většině specifikací programovacích jazyků. V této pasáži si na příkladu zápisu čísel ukážeme základní pravidla interpretace syntaktických diagramů.

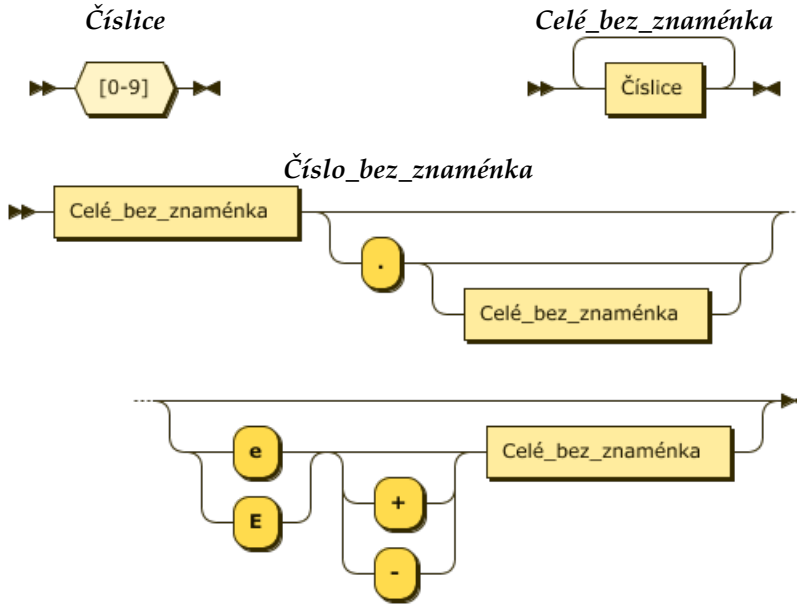
Syntaktické diagramy interpretujeme tak, že jakákoliv cesta od počáteční značky ⏩ (vstupního bodu) po koncovou značku ⋈ (výstupního bodu) generuje syntakticky korektně zapsanou konstrukci jazyka. Pokud existují nějaké výjimky, bývají popsané v doprovodném textu. Cestou procházíte elementy, které budou jednoho ze tří druhů:

- Šestiúhelník obsahuje specifikaci zapsanou přímo v EBNF. Používá se spíše výjimečně, a to ve chvíli, kdy je tento zápis úspornější než odpovídající diagram, aniž by ztratil na své přehlednosti. Zápis v diagramu prvku *Číslice* na obrázku B.1 např. označuje, že za číslici považujeme jakýkoliv znak od nuly do devítky.
- Obdélník označuje prvek, jenž je definován v jiném syntaktickém diagramu.
- Obdélník se zaoblenými rohy obsahuje znak či skupinu znaků, která se má na daném místě vložit (zapsat) do vytvářeného kódu.

Elementy jsou navzájem pospojovány spojnicemi určujícími možné cesty od prvku k prvku při skládání dané syntaktické konstrukce.

[58] EBNF je zkratka termínu *Extended Backus-Naur form* (*Rozšířená Backus-Naurova forma*), což je jazyk používaný k definici syntaxe programovacích jazyků.

Syntaxe zápisu čísel je specifikována prostřednictvím tří syntaktických diagramů na obrázku B.1. Symbol v diagramu ***Číslice*** symbolizuje, že číslicí může být libovolný znak od `0` do `9`. Diagram ***Celé_bez_znaménka*** prozrazuje, že tento druh entity je tvořen nejméně jednou číslicí, ale že číslic může být i více. Diagram ***Číslo_bez_znaménka*** pak specifikuje všechny povolené způsoby, jakými lze zapsat číslo.



Obrázek B.1:
Syntaktický diagram zobrazující pravidla pro zápis čísel

Příloha C
Konvence pro psaní programů v Pythonu

Oficiální konvence jsou včetně demonstračních ukázek podrobně popsány v dokumentu PEP 8,[59] který najdete na adrese https://www.python.org/dev/peps/pep-0008/, a PEP 257, který najdete na adrese https://www.python.org/dev/peps/pep-0257/. První probírá konvence pro psaní kódu, druhý konvence pro psaní dokumentačních komentářů. Zde uvedu jen stručný výčet.

Zájemcům, kteří chtějí rychle ověřit, že nějaký kód vyhovuje těmto konvencím, doporučuji, aby si na adrese https://pep8.readthedocs.io stáhli program nazvaný příhodně `pep8`, jenž dodržování těchto konvencí ověří za vás.

Uspořádání kódu

Hlavní zásadou při tvorbě kódu by měl být fakt, že program se daleko častěji čte, než zapisuje, takže bychom jej měli zapisovat tak, aby se následně dobře četl. Bude-li kód čitelnější, když v daném místě porušíte některou z konvencí, porušte ji. To je obzvláště důležité u programů určených pro výuku.

- Odsazujte o 4 znaky a v textu nepoužívejte tabulátory, ale jen mezery.
- Omezte délku řádků na 79 znaků, u komentářů (i dokumentačních) na 72 znaků.
- U delších řádků vkládejte konec řádku před binární operátor, takže výraz bude na dalším řádku začínat operátorem – např.

```
suma = (první proměnná
        + druhá proměnná)
```

[59] PEP je zkratka z anglického *Python Enhancement Proposal* (doporučení pro vylepšení *Pythonu*). Jsou to dokumenty poskytující informace komunitě *Pythonu* nebo popisující nové funkce, procesy nebo prostředí. Jejich přehled najdete na https://www.python.org/dev/peps/.

- Definice tříd a samostatných funkcí oddělujte dvěma řádky, definice metod uvnitř třídy oddělujte jedním prázdným řádkem.
- Používejte kódování UTF-8, a to bez občas nabízené úvodní deklarace.
- Vkládejte každý import modulu na samostatný řádek, import několika identifikátorů z daného modulu (`from … import …`) je však možné uvést společně.
- Umístěte importy na počátek modulu před definice globálních proměnných.
- Nepoužívejte hvězdičkový import (`from … import *`).
- Identifikátory uvozené a ukončené dvojicí podtržení (tzv. *dunders* jako zkratka z anglického ***d****ouble* ***unders****cores*) by měly být definovány na počátku modulu za dokumentačním komentářem, ale před importy.
- Pro trojité ohraničení stringů používejte trojité uvozovky, abyste byli konzistentní s dokumentačními komentáři podle PEP 257. Jednoduchý ohraničující znak (apostrof či uvozovky) používejte dle svých preferencí, ale buďte konzistentní.
- Mažte závěrečné mezery na koncích řádků. (Některé editory tuto funkci nabízejí.)
- U pojmenovaných argumentů nepoužívejte mezery kolem znaku `=`.
- Nevkládejte více příkazů na jeden řádek.
- U složených příkazů pokračujte na řádku za hlavičkou pouze v případě, že tělo tvoří jeden jednoduchý příkaz – např.

```
for x in lst: total += x
```

- Nebojte se použít závěrečné čárky v seznamech argumentů, mohou usnadnit přidání dalšího členu – např.

```
FILES = [
    'setup.cfg',
    'tox.ini',
    ]
```

- Komentáře, které neodpovídají kódu, jsou horší než žádné. Udržujte komentáře neustále aktuální (*up-to-date*).

Jmenné konvence

Jmenné konvence bohužel nejsou zcela konzistentní, nicméně existuje několik doporučení, které by měly nové programy dodržovat.

- Ve standardní knihovně musí všechny identifikátory používat pouze znaky `ASCII` a měly by pokud možno používat angličtinu. Doporučuje se toto pravidlo dodržovat i v ostatních programech.
- Názvy viditelné uživateli jako součást API by měly reflektovat spíše užití než implementaci.
- Vyvarujte se názvů tvořených pouze znakem `l` (malé L), `O` (velké o) nebo `I` (velké i). Některé fonty neumí odlišit znaky `1-l-I` (jedna – L – I) a `0-O` (nula – o).

- Používejte ve svém vývojovém prostředí font, který tyto znaky odliší.
- Názvy modulů by měly být krátké a používat jen malá písmena. Použití znaku podtržení se nedoporučuje.
- Názvy tříd by měly používat `VelbloudíNotaci`.
- Názvy funkcí by měly používat jen malá písmena a jednotlivá slova názvu oddělovat podtržítky – např. `long_function_name`.
- První parametr instančních metod by se měl vždy jmenovat `self`.
- První parametr třídních metod by se měl vždy jmenovat `cls`.
- Názvy neveřejných metod a instančních proměnných by měly začínat podtržítkem.
- Názvy konstant by měly používat pouze velká písmena a jednotlivá slova názvu oddělovat podtržítky – např. `LONG_CONSTANT_NAME`.
- U každého atributu (proměnné i metody) se vždy rozmyslete, zda má být veřejný. Neveřejný atribut lze snadno zveřejnit, obrácená operace je po rozšíření dané třídy či modulu již zakázaná.

Dokumentační komentáře (PEP 257)

Dokumentační komentář je stringový literál zadaný jako první příkaz v modulu, třídě, metodě či funkci. Ten se automaticky stane hodnotou atributu `__doc__` daného objektu.

- Dokumentační komentář by měly mít všechny moduly a všechny funkce a třídy exportované z těchto modulů.
- Dokumentační komentář balíčku je možné zadat v jeho souboru `__init__.py`.
- Dokumentační komentáře ohraničujte vždy trojitými uvozovkami (`"""`), s případnou předponou `r`, používáte-li v nich zpětná lomítka.
- Trojité uvozovky používejte, i když se komentář vejde na řádek, jak je tomu např. ve výpisu 38.2 na straně 559. Neoddělujte ohraničení mezerami.
- Víceřádkové komentáře zahajte jednořádkovým shrnutím umístěným na stejném řádku s ohraničujícími uvozovkami. Protože může být použito automatickými indexačními programy, je důležité, aby bylo na jednom řádku a aby bylo od dalšího textu odděleno prázdným řádkem. Další řádky komentáře zarovnávejte stejně jako uvozovky na prvním řádku.
- Za dokumentačním komentářem třídy vynechte řádek.
- Dokumentační komentář modulu by měl vypsat všechny třídy, výjimky a funkce daného modulu.
- Dokumentační komentář třídy by měl vypsat její chování, seznam veřejných metod a atributů včetně instančních.
- Dokumentační komentář funkce by měl vypsat její funkci, argumenty a návratovou hodnotu, vedlejší efekty a případná omezení.

Příloha D Stručná historie posledních verzí

Jak jsem již řekl v pasáži *Potřebné vybavení* na straně 29, pokusím se knihu psát tak, aby ji mohli používat i uživatelé starších verzí než verze 3.10, která je uvedena v podtitulu. Nebudu se ale dívat příliš zpět. Kniha předpokládá, že všichni mají instalovánu minimálně verzi 3.6, která vyšla v prosinci 2016 a která je velmi často deklarována jako povinný základ. Následující přehled stručně připomíná verze, jež vyšly v posledních letech. Přidaných vlastností a rysů je samozřejmě mnohem více, uvádím jen ty, které se přímo týkají probíraných konstrukcí. Kompletní přehled najdete v dokumentaci, která je součástí standardní instalace (viz pasáž *Dokumentace* na straně 35).

- Verze 3.6 – prosinec 2016:
  - F-stringy, např. `f'Proměnná x = {x}'`.
  - Anotace proměnných, např. `atr:str`.
  - Podtržení pro zpřehlednění čísel, např. `123_456_789`.
  - Konzole *Windows* akceptuje UTF-8.
  - Slovník iteruje v pořadí přidání.
- Verze 3.7 – červen 2018:
  - Klíčová slova `async` a `await`.
  - Zrušení `ASCII` jako implicitního kódování textu.
  - Příprava odloženého vyhodnocování anotací.
  - Funkce mohou mít více než 255 argumentů.
- Verze 3.8 – listopad 2019:
  - Přiřazovací výraz, např. `pi:=3.14`.
  - Pouze poziční parametry, např. `divmod(x, y, /)`.
  - Přiřazení v nahrazovacích polích f-stringů, např. `f'Proměnná {x=}'`.

- Verze 3.9 – říjen 2020:
  - Slovníky akceptují operátor **|** a **|=**, např. `d |= {k1:v1, k2:v2}`.
  - V anotacích lze používat deklarace kontejnerů standardních typů, např. `iarr:tuple[int]`.
  - Stringům přibyly metody pro odstranění předpon a přípon.
  - Uvolnění gramatických omezení pro dekorátory.
  - Přidána třída `typing.Annotated` pro rozšíření možností anotací.
- Verze 3.10 – říjen 2021:
  - Zavedena měkká klíčová slova.
  - V příkazu `with` je nyní možné použít závorky, a tím umožnit rozložení hlavičky na více řádků.
  - Upřesnění hlášení o syntaktické chybě.
  - Zavedení přepínače `match … case`.
  - Vyhodnocování anotací je implicitně odloženo; anotace se nyní ukládají jako stringy.
  - Datové typy lze nyní sjednocovat, např. anotace `number:int|float` deklaruje, že proměnná `number` bude buď celé, nebo reálné číslo.
  - Zaveden datový typ `typing.Calleable` umožňující deklarovat typy parametrů volatelných objektů.
  - Zavedena anotace `TypeAlias` označující, že anotovaná proměnná bude anotací – hovoří se o anotacích přezdívek typů (TypeAlias Annotation).

Část G
Seznamy

Tato část obsahuje seznamy hypertextových odkazů na výpisy programů, obrázky, tabulky a odbočky – podšeděné bloky. Není součástí tištěné knihy, aby zbytečně nezvyšovala její cenu. Je však součástí elektronických verzí, v nichž můžete hypertextové odkazy efektivně využít.

Seznam výpisů programů

Výpis 1.1: Komunikace s interpretem spuštěným v konzolovém okně 39
Výpis 1.2: Reakce na odsazení příkazu....43
Výpis 2.1: Reakce na zadání velkého čísla 52
Výpis 2.2: Zpřehlednění čísel vložením podtržítek 52
Výpis 2.3: Zadávání a zobrazování desetinných čísel 53
Výpis 2.4: Zadávání a zobrazování komplexních čísel 53
Výpis 2.5: Zápis čísel v alternativních číselných soustavách....54
Výpis 2.6: Reakce na použití objektů `None` a `...` (`Ellipsis`)....56
Výpis 3.1: Zápis jednořádkového textu 58
Výpis 3.2: Zápis víceřádkového textu 59
Výpis 3.3: Zadání znaků pomocí escape sekvencí 61
Výpis 3.4: Komentáře....63
Výpis 3.5: Zadání na více řádcích....63
Výpis 4.1: Zadání argumentu nápovědy pomocí stringu....78
Výpis 4.2: Přepnutí do režimu nápovědy a zpět 79
Výpis 4.3: Volání funkce rozepsané na více řádcích 80
Výpis 5.1: Ukázky tří druhů dělení 86
Výpis 5.2: Komplexní číslo jako exponent – Eulerova rovnost 87
Výpis 5.3: Nekonečna a nesmyslná čísla 88
Výpis 5.4: Sčítání textů 89
Výpis 5.5: Násobení textů....89
Výpis 5.6: Indexace stringů....90
Výpis 6.1: Přiřazení hodnoty proměnné....96
Výpis 6.2: Skrytá běhová chyba....97
Výpis 6.3: Současné přiřazení skupiny hodnot proměnným 98
Výpis 6.4: Zavedení proměnných `inf` a `nan`....99
Výpis 6.5: Ukázky vnořeného volání funkcí 100
Výpis 6.6: Zjištění typu objektu 101
Výpis 6.7: Přiřazení funkcí (přesněji odkazů na funkce) do proměnných 102
Výpis 6.8: Definice a použití lambda-výrazu 103
Výpis 6.9: Přiřazovací výraz a jeho použití 106
Výpis 6.10: Ovlivnění toho, bude-li se zobrazovat systémový či uživatelský podpis 108
Výpis 7.1: Převody logických hodnot na čísla a jiných hodnot na logické 110
Výpis 7.2: Zřetězené porovnávání 112
Výpis 7.3: Test totožnosti objektů 113
Výpis 7.4: Chování logických operátorů....114
Výpis 7.5: Vliv priorit na výsledek 115
Výpis 7.6: Chování bitových operátorů 116
Výpis 7.7: Bitové posuny....117
Výpis 7.8: Použití podmíněného výrazu....118

Výpis 8.1: Složené přiřazovací operátory ... 122
Výpis 8.2: Použití příkazu `del` ... 123
Výpis 8.3: Použití příkazu `assert` ... 125
Výpis 8.4: Spuštění systému Python s aktivovaným a deaktivovaným příkazem `assert` ... 126
Výpis 8.5: Kontrola shodnosti odsazení příkazů ... 126
Výpis 9.1: Importování modulů ... 134
Výpis 9.2: Přejmenování importovaného modulu ... 135
Výpis 9.3: Import zadaných objektů ze zadaného modulu ... 137
Výpis 9.4: Použitelnost proměnné s odkazem na modul ... 138
Výpis 9.5: Import všech objektů ze zadaného modulu ... 139
Výpis 9.6: Demonstrace nárůstu počtu atributů po hromadném importu ... 139
Výpis 9.7: Modul `builtins` ... 142
Výpis 10.1: Modul `m10a_Modul` sloužící k demonstraci chování Pythonu při zavádění modulu ... 145
Výpis 10.2: Import modulu `m10a_Modul` ... 150
Výpis 10.3: Nové načtení modulu `m10a_Modul` ... 154
Výpis 10.4: „Neaktualizace“ přímo importovaných odkazů ... 154
Výpis 10.5: Opětné načtení verze s chybou ... 155
Výpis 10.6: Definice modulu `m10b_Demo_all` vyjmenovávajícího veřejné atributy ... 157
Výpis 10.7: Hvězdičkový import modulu `m10b_Demo_all` ... 158
Výpis 10.8: Import modulu `posixpath` nebo `ntpath` zprostředkovaný modulem `os` ... 159
Výpis 10.9: Prověrka zprostředkování importu ... 159
Výpis 10.10: Modul `m10c_Circular_C` importující modul `m10d_Circular_D` ... 160
Výpis 10.11: Modul `m10d_Circular_D` importující modul `m10c_Circular_C` ... 160
Výpis 10.12: Pokus o import modulu `m10c_Circular_C` a reakce na něj ... 161
Výpis 11.1: Jednořádková definice funkce ... 164
Výpis 11.2: Víceřádková definice funkce ... 165
Výpis 11.3: Přisazení definice funkce v interaktivním režimu ... 168
Výpis 11.4: Modul `m11a_Funkce_v_modulu` demonstrující pravidla pro definice funkcí v modulu ... 169
Výpis 11.5: Dynamická povaha jazyka při definici a volání funkcí ... 170
Výpis 11.6: Definice modulu `m11b_dbg_demo` ... 172
Výpis 11.7: Import modulu `m11b_dbg_demo` a zavolání jeho funkce `fce1()` ... 173
Výpis 12.1: Definice a volání funkce s parametrem ... 175
Výpis 12.2: Definice funkce s lokálními proměnnými ... 176
Výpis 12.3: Volání funkcí s parametry ... 176
Výpis 12.4: Funkce s povinně pojmenovanými argumenty ... 177
Výpis 12.5: Funkce s povinně pozičními argumenty ... 178
Výpis 12.6: Funkce s různými kombinacemi povinných parametrů ... 180
Výpis 12.7: Volání funkce `print()` s vlastní hodnotou implicitního argumentu ... 181
Výpis 12.8: Definice funkce se dvěma implicitními argumenty a možnosti jejího volání ... 182
Výpis 12.9: Konstantnost předdefinovaných hodnot ... 182
Výpis 12.10: Funkce s vedlejším efektem ... 183
Výpis 12.11: Definice a použití funkcí vracejících hodnotu ... 184
Výpis 12.12: Nepoužitelnost přetěžování funkcí ... 184
Výpis 12.13: Deklarace typů proměnných a návratových hodnot prostřednictvím anotací ... 186
Výpis 13.1: Definice vnitřních funkcí ... 188
Výpis 13.2: Zakrytí globální proměnné stejnojmennou lokální ... 192
Výpis 13.3: Použití příkazu `global` ... 193
Výpis 13.4: Použití příkazu `nonlocal` ... 195

Výpis 13.5: Demonstrace vnořeného volání funkcí ... 197
Výpis 13.6: Použití funkce vracející funkci ... 198
Výpis 13.7: Využití atributů funkce ... 199
Výpis 13.8: Nelokální proměnné a uzávěry ... 200
Výpis 13.9: Definice modulu `m13b_Circular_B` importujícího modul `m13c_Circular_C` ... 201
Výpis 13.10: Definice modulu `m13c_Circular_C` importujícího modul `m13b_Circular_B` ... 201
Výpis 13.11: Import modulu `m13b_Circular_B` ... 202
Výpis 14.1: Ukázka použití jednoduchého podmíněného příkazu ... 204
Výpis 14.2: Ukázka použití úplného podmíněného příkazu ... 205
Výpis 14.3: Ukázky použití rozšířeného podmíněného příkazu ... 207
Výpis 14.4: Řešení jednoduché úlohy prostřednictvím přepínače ... 208
Výpis 14.5: Ukázka možného sdružování hodnot ve vzorech ... 209
Výpis 14.6: Přímé zadání podmíněného příkazu v interaktivním režimu ... 210
Výpis 15.1: Ukázky definic funkcí používajících rekurzivní volání ... 212
Výpis 15.2: Rekurzivní definice faktoriálu ... 213
Výpis 15.3: Ukázky definic funkcí používajících cyklus `while` ... 214
Výpis 15.4: Nekonečný cyklus ... 215
Výpis 15.5: Použití příkazu `break` ... 216
Výpis 15.6: Použití přiřazovacího výrazu v hlavičce cyklu `while` ... 218
Výpis 15.7: Cyklus `while` využívající větev `else` ... 219
Výpis 15.8: Použití příkazu `continue` ... 220
Výpis 15.9: Použití příkazu `for` se zdrojem definovaným jako sada hodnot ... 221
Výpis 15.10: Použití příkazu `for` se zdrojem generovaným funkcí `range()` ... 222
Výpis 15.11: Indexace znaků v textovém řetězci – stringu ... 223
Výpis 15.12: Zpracování jednotlivých znaků v textovém řetězci bez použití indexů ... 223
Výpis 15.13: Ukázka zanořování cyklů ... 224
Výpis 15.14: Příkaz `for` s postupným použitím více zdrojů ... 225
Výpis 15.15: Příkaz `for` s větví `else` ... 225
Výpis 16.1: Ukázky syntaktických chyb a reakcí na ně při zadávání příkazů v prostředí IDLE ... 228
Výpis 16.2: Ukázky syntaktických chyb a reakcí na ně při zadávání příkazů z konzolového okna ... 230
Výpis 16.3: Zopakovaný výpis 10.5 ze strany 157 zobrazující načtení třetí verze modulu `m10a_Modul` z výpisu 10.1 na straně 147 ... 231
Výpis 16.4: Ukázka zachycení a ošetření výjimky ... 234
Výpis 16.5: AHA-příklad používající příkaz `try` se všemi dostupnými větvemi ... 238
Výpis 16.6: Měření rychlosti počítače a přesnosti odhadu času ... 239
Výpis 16.7: Zdánlivé záludnosti větve `finally` ... 240
Výpis 16.8: Funkce `assert_stmt()` demonstrující funkci příkazu `assert` ... 242
Výpis 17.1: Demonstrace neměnnosti objektů ... 245
Výpis 17.2: Vytváření seznamů pomocí literálů ... 247
Výpis 17.3: Vytváření seznamů pomocí konstruktoru `list()` ... 248
Výpis 17.4: Vytváření seznamů z jiných seznamů pomocí sčítání a násobení ... 249
Výpis 17.5: Vytvoření seznamu pomocí generátorové notace ... 250
Výpis 17.6: Použití funkcí `append()` a `extend()` ... 252
Výpis 17.7: Demonstrace chování proměnných objektů ... 253
Výpis 17.8: Postupné budování seznamu ... 254
Výpis 17.9: Rozšiřování seznamu přičítáním ... 254
Výpis 17.10: Chybné pokusy o rozšíření seznamu ... 255
Výpis 17.11: Ukázky použití modifikačních funkcí a metod pracujících s indexy ... 256

Výpis 17.12: Ukázky použití modifikačních metod pracujících s celým seznamem....258
Výpis 17.13: Modul `m17a_Pascal` s funkcemi pracujícími s vícerozměrnými seznamy....259
Výpis 17.14: Načtení modulu `m17a_Pascal` a prověření jeho funkce....259
Výpis 17.15: Anotace odkazující na seznamy....260
Výpis 18.1: Vytváření n-tic pomocí literálů....262
Výpis 18.2: Vytváření n-tic pomocí konstruktoru `tuple()`....264
Výpis 18.3: Vytváření n-tic z jiných n-tic pomocí sčítání a násobení....264
Výpis 18.4: Vytváření n-tic přičítáním....265
Výpis 18.5: Balení a rozbalování n-tic a seznamů....266
Výpis 18.6: Prohazování hodnot proměnných....267
Výpis 18.7: Hvězdičkové pravidlo....267
Výpis 18.8: Vytvoření n-tice pomocí generátoru....268
Výpis 18.9: Přístup k prvkům n-tic....270
Výpis 18.10: Sčítání seznamů a n-tic....270
Výpis 18.11: Chování n-tic s proměnnými a neměnnými prvky....271
Výpis 18.12: Prvky standardních n-tic nemají jména....272
Výpis 18.13: Pojmenované n-tice....272
Výpis 18.14: Pojmenované n-tice s implicitními argumenty....273
Výpis 19.1: Vytváření množin pomocí literálů....276
Výpis 19.2: Vytváření množin pomocí konstruktoru `set()`....277
Výpis 19.3: Povolené a nepovolené argumenty konstruktoru `set()`....277
Výpis 19.4: Vytváření množin prostřednictvím množinových operací....278
Výpis 19.5: Vytváření množin pomocí generátorové notace....280
Výpis 19.6: Vytváření zmrazených množin....281
Výpis 19.7: Jednoprvkové modifikační množinové operace....282
Výpis 19.8: Modifikace množin množinami či jinými zdroji....283
Výpis 19.9: Porovnání množin....285
Výpis 20.1: Vytváření slovníků pomocí literálů....287
Výpis 20.2: Vytváření slovníků pomocí konstruktoru `dict()`....288
Výpis 20.3: Vytváření slovníků pomocí funkce `fromkeys()`....290
Výpis 20.4: Vytváření slovníků pomocí generátorové notace....290
Výpis 20.5: Přístup k položkám prostřednictvím klíčů....292
Výpis 20.6: Metody pro práci s jednotlivými položkami....293
Výpis 20.7: Použití metody `update()`....294
Výpis 20.8: Pohledy....295
Výpis 20.9: Pohledy jako generátory....296
Výpis 20.10: Operace s pohledy....297
Výpis 21.1: Demonstrace chování programu při předávání argumentů hodnotou....299
Výpis 21.2: Definice a ukázky použití funkce `gr()`....300
Výpis 21.3: Zpracování argumentů při použití hvězdičkového parametru....302
Výpis 21.4: Použití a zpracování hvězdičkového argumentu....303
Výpis 21.5: Zpracování argumentů při použití dvouhvězdičkového parametru....304
Výpis 21.6: Zpracování dvouhvězdičkového argumentu....305
Výpis 21.7: Příklady zpracování argumentů inspirované [5]....306
Výpis 21.8: Možné použití hvězdičkových a dvojhvězdičkových argumentů....307
Výpis 22.1: Použití formátovacího operátoru `%`....313
Výpis 22.2: Použití prázdné formátové položky....315
Výpis 22.3: Různé způsoby zadávání nahrazovaného textu....317
Výpis 22.4: Efekt zadání konverze konverzí....318
Výpis 22.5: Zadání požadovaného minimálního počtu pozic....319

Výpis 22.6: Zadání požadované přesnosti, resp. maximálního počtu pozic 320
Výpis 22.7: Typy formátů pro celočíselné hodnoty 321
Výpis 22.8: Typy formátů pro zobrazení reálných čísel 322
Výpis 22.9: Použití oddělovače skupin číslic 323
Výpis 22.10: Zadání oddělovačů skupin číslic a alternativního formátu 324
Výpis 22.11: Zarovnání a plnění 326
Výpis 22.12: Používání vnořených nahrazovacích polí 327
Výpis 22.13: Definice funkce `printpasf()` v modulu `m22a_PrintPasF` 328
Výpis 22.14: Použití funkce `printpasf()` v modulu `m22a_PrintPasF` 329
Výpis 22.15: Trasovací funkce `prSE()` v modulu `dbg` 330
Výpis 22.16: Trasovací funkce `prIN()` v modulu `dbg` 331
Výpis 22.17: Ukázka použití trasovacích funkcí `prSE()` a `prIN()` 332
Výpis 23.1: Hluboké a mělké kopie 335
Výpis 23.2: Použití operátorů `in` a `not in` 338
Výpis 23.3: Použití funkce `reversed()` 339
Výpis 23.4: Použití funkce `sorted()` 339
Výpis 23.5: Indexování a vykrajování u rozsahů – objektů typu `range` 341
Výpis 23.6: Modifikace obsahu posloupnosti 342
Výpis 23.7: Postup při výměně prvku v n-tici 343
Výpis 24.1: Zjištění a změna aktuální složky 349
Výpis 24.2: Syntéza a analýza cest 351
Výpis 24.3: Vytváření a mazání složek 352
Výpis 24.4: Získání informací o souborech 353
Výpis 24.5: Otevření souboru 355
Výpis 24.6: Zápis dat, splachování a zavírání souborů 356
Výpis 24.7: Čtení souborů 359
Výpis 26.1: Definice třídy a jejích atributů 375
Výpis 26.2: Práce s atributy objektu 377
Výpis 26.3: Přidávání a odstraňování atributů objektu 377
Výpis 26.4: Přidávání a odstraňování metody 378
Výpis 26.5: Třída jako parametr funkce 379
Výpis 26.6: Vytvoření prvních instancí 380
Výpis 26.7: Kvalifikace atributu třídy instancí 381
Výpis 26.8: Možná použití dekorátoru 383
Výpis 26.9: Přidání a použití instanční metody 384
Výpis 26.10: Definice třídy `C2` 385
Výpis 26.11: Vytváření instancí třídy `C2` a používání jejich atributů 387
Výpis 26.12: Vytváření nových instančních datových atributů 388
Výpis 26.13: Změny třídních datových atributů 389
Výpis 26.14: Třídní datový atribut jako implicitní hodnota instančního 390
Výpis 26.15: Zavádění nových třídních funkčních atributů 391
Výpis 26.16: Zavádění nových instančních funkčních atributů 391
Výpis 26.17: Demonstrace nutnosti kvalifikace atributů 393
Výpis 26.18: Ukázky použití systémových atributů 395
Výpis 27.1: Definice tříd `LA`, `LB` a `LC` v modulu `m27a_LABC` 399
Výpis 27.2: Jmenné prostory a dosažitelné atributy tříd z modulu `m27a_LABC` 401
Výpis 27.3: Jmenné prostory a dosažitelné atributy instancí tříd z modulu `m27a_LABC` 402
Výpis 27.4: Volání metod v hierarchii dědění 403
Výpis 27.5: Demonstrace dědění initorů 404

Výpis 27.6: Ukázka definice a použití vlastní výjimky ... 406
Výpis 28.1: Zdrojový kód tříd z modulu m28a_Diamant demonstrujících chování tříd uspořádaných do diamantové architektury ... 409
Výpis 28.2: Prověrka vlastností násobného dědění ... 410
Výpis 28.3: Definice třídy Dcera2 v modulu m28a_Diamant ... 412
Výpis 28.4: Použití třídy Dcera2 ... 412
Výpis 28.5: Definice modulu m28b_Arguments s třídami používajícími násobné dědění a konstruktory s parametry ... 413
Výpis 28.6: Průběh inicializace instance třídy DceraA ... 414
Výpis 28.7: Nerealizovatelná a následně opravená hierarchie tříd ... 416
Výpis 28.8: Definice složitější hierarchie tříd ... 416
Výpis 28.9: Pseudosoukromý atribut a komolení jmen ... 418
Výpis 28.10: Definice tříd Bc a Mc v modulu m28c_Komolení ... 419
Výpis 28.11: Použití tříd Bc a Mc a jejich instancí ... 420
Výpis 29.1: Přímá definice vlastností jako instance třídy property ... 423
Výpis 29.2: Definice vlastností pomocí lambda-výrazů ... 425
Výpis 29.3: Důsledky změny třídního atributu definujícího vlastnost ... 426
Výpis 29.4: Zadání vlastnosti pomocí dekorátoru ... 427
Výpis 29.5: Demonstrace použití vlastností instancí třídy C29c ... 428
Výpis 29.6: Definice formálně abstraktní třídy Matka29a bez abstraktních metod ... 430
Výpis 29.7: Definice abstraktní třídy Matka29b s abstraktní metodou ... 431
Výpis 29.8: Definice potomka abstraktní třídy nepřebíjejícího abstraktní metodu ... 432
Výpis 29.9: Definice třídy Dcera29b, jež je potomkem abstraktní třídy Matka29b ... 432
Výpis 29.10: Definice dalších druhů abstraktních metod ... 434
Výpis 30.1: Vytvoření výčtového datového typu prostřednictvím volání funkce Enum() a demonstrace vlastností daného typu a jeho atributů–instancí ... 439
Výpis 30.2: Definice výčtového datového typu jako třídy ... 441
Výpis 30.3: Definice výčtového typu Směr demonstrujícího některé další možnosti ... 442
Výpis 30.4: Automatické přiřazování obalovaných hodnot ... 443
Výpis 30.5: Poloautomatické nastavování obalovaných hodnot ... 444
Výpis 30.6: Definice datové třídy a její test ... 446
Výpis 30.7: Získání hodnot v rámci vyhodnocování vzorů v příkazu match ... 448
Výpis 30.8: Prověřování posloupností ... 449
Výpis 30.9: Použití vzoru s výběrem hodnot ... 450
Výpis 31.1: Šablona initorů standardních balíčků uložená v souboru __init__.py v kořenovém balíčku ... 453
Výpis 31.2: Výkonný kód modulu m1_cls v balíčku PKG.sub_pkg.subsub_pkg ... 453
Výpis 31.3: Import modulu PKG.sub_pkg.subsub_pkg.m1_cls a jeho opětné načtení po úpravě ... 454
Výpis 31.4: Kód modulu PKG.sub_pkg.subsub_pkg.m2_dot bez úvodních a závěrečných tisků ... 455
Výpis 31.5: Import modulu PKG.sub_pkg.subsub_pkg.m2_dot ... 456
Výpis 31.6: Kód modulu PKG.sub_pkg.subsub_pkg.m3_star, tentokrát včetně úvodních a závěrečných tisků ... 457
Výpis 31.7: Průběh m3_star run() ... 458
Výpis 31.8: Obsah souboru __init__.py v balíčku PKG.sub_pkg.subsub_pkg ... 459
Výpis 31.9: Průběh znovuzavedení aktualizovaného balíčku PKG.sub_pkg.subsub_pkg ... 459
Výpis 31.10: Průběh znovuzavedení aktualizovaného balíčku m3_star ... 460

Výpis 31.11: Definice modulu mA_importing v balíčku NS.ns_pkg ve standardním zdroji....463
Výpis 31.12: Definice modulu mB_imported v balíčku NS.ns_pkg v externím zdroji....463
Výpis 31.13: Zavedení modulu mA_importing....464
Výpis 31.14: Definice souboru __init__ – initoru balíčku NS.pkg ve stromu 68_NS....466
Výpis 31.15: Import atributu balíčku NS.pkg....466
Výpis 32.1: Definice modulu m32a_Script....468
Výpis 32.2: Reakce na spuštění modulu m32a_Script v interaktivním režimu....469
Výpis 32.3: Reakce na spuštění modulu m32a_Script v příkazovém okně Windows....469
Výpis 32.4: Modul m4_run v balíčku PKG.sub_pkg.subsub_pkg....470
Výpis 32.5: Vytvoření samostatné aplikace pomocí modu winapp a její spuštění....471
Výpis 32.6: Definice modulu __main__ v souboru PKG.pyz....471
Výpis 32.7: Definice funkce main() v modulu m32b_APR....473
Výpis 32.8: Spouštění skriptu m32b_APR....474
Výpis 33.1: Definice třídy Prime, její instance jsou iterovatelné objekty a současně iterátory....480
Výpis 33.2: Generátorový výraz a jeho použití....481
Výpis 33.3: Nekorektní použití generátorového výrazu....482
Výpis 33.4: Definice a použití generátorové funkce prime_fun()....484
Výpis 33.5: Definice generátorové funkce gen1() demonstrující vícenásobné použití příkazu yield....485
Výpis 33.6: Demonstrace prostřednictvím ruční aktivace výrazu yield....488
Výpis 33.7: Definice generátorové funkce ord_gen() pracující s výrazem yield....490
Výpis 33.8: Definice funkce orders() využívající možnost ovlivnit generátor....491
Výpis 33.9: AHA-příklad s definicí generátorových funkcí gen2() a gen21() demonstrujících použití výrazů yield a yield from....493
Výpis 34.1: Využití dekorátoru total_ordering pro doplnění porovnávacích metod....497
Výpis 34.2: Definice třídy Zlomek v modulu m34a_Zlomky....501
Výpis 34.3: Práce s instancemi třídy Zlomek....502
Výpis 34.4: Definice třídy Indexer přetěžující operátor []....504
Výpis 34.5: Přidání instanční metody pro konkrétní instanci....505
Výpis 34.6: Demonstrace rozdílů v kódu při použití příkazu with....507
Výpis 34.7: Emulace postupu při zadání příkazu with....508
Výpis 35.1: Anotace parametrů funkce, její návratové hodnoty a atributů modulu....512
Výpis 35.2: Zapamatované anotace při okamžitém vyhodnocení platné doposud....513
Výpis 35.3: Zapamatované anotace při odloženém vyhodnocení plánované od verze 3.11....514
Výpis 35.4: Co o anotacích prozradí nápověda....515
Výpis 35.5: Definice a použití přezdívek datových typů....517
Výpis 35.6: Slučování datových typů....518
Výpis 36.1: Ukázka použití třídních metod....522
Výpis 36.2: Ukázka definice tovární metody....524
Výpis 36.3: Definice dekorátoru – třídy traced v modulu m36a_Traced....525
Výpis 36.4: Použití dekorátoru traced....526
Výpis 36.5: Definice dekorátorů jedináčka v modulu m36c_Singleton....529
Výpis 36.6: Použití dekorátoru singleton....530
Výpis 37.1: Ekvivalent definice vestavěné třídy property v modulu m37a_Property....533
Výpis 37.2: Definice třídy descriptor....534
Výpis 37.3: Hrátky s deskriptorem....536
Výpis 37.4: Postup metody __getattribute__() při vyhodnocování atributu instance....538

Výpis 37.5: Deskriptor pro odložené vyhodnocení ... 539
Výpis 37.6: Definice třídy C37c v modulu m37c_Atributy ... 542
Výpis 37.7: Ukázky reakcí na přístup k atributům ve třídě C37c ... 543
Výpis 37.8: Definice třídy C37d v modulu m37c_Atributy ... 546
Výpis 37.9: Práce s atributy instancí třídy C37d ... 547
Výpis 37.10: Definice modulu m37d_ModuleProperty s vlastní mateřskou třídou ... 548
Výpis 37.11: Import modulu m37d_ModuleProperty a práce s jeho vlastností ... 549
Výpis 37.12: Ukázka používání slotů ... 551
Výpis 38.1: Použití metody __init_subclass__() ... 553
Výpis 38.2: Šablona testu funkce metatřídy v modulu m38a_TestMetaXxx ... 559
Výpis 38.3: Import modulu m38a_TestMetaXxx ... 559
Výpis 38.4: Definice metatřídy MetaFce definované jako funkce v modulu m38b_MetaFce ... 560
Výpis 38.5: Import modulu m38b_MetaFce ... 561
Výpis 38.6: Definice metatřídy MetaCls definované jako třída v modulu m38c_MetaCls ... 561
Výpis 38.7: Import modulu m38c_MetaCls ... 562
Výpis 38.8: Definice metatřídy MetaObj definované v modulu m38d_MetaObj jako objekt ... 563
Výpis 38.9: Import modulu m38d_MetaObj ... 564
Výpis 38.10: Definice jedináčka pomocí __new__() ... 565
Výpis 38.11: Definice jedináčka pomocí metatřídy ... 566
Výpis 39.1: Definice a použití jednoduchých korutin ... 570
Výpis 39.2: Zabalení korutiny do instance třídy Task ... 571
Výpis 39.3: Analýza datových typů ... 573

Seznam obrázků

Obrázek 1.1: Okno s dokumentací platformy 35
Obrázek 1.2: Okno příkazového řádku Windows se spuštěným interpretem Pythonu 39
Obrázek 1.3: Okno vývojového prostředí IDLE 41
Obrázek 1.4: Okno vývojového prostředí IDLE 42
Obrázek 4.1: Pokus o výpis rozsáhlejší nápovědy v prostředí IDLE 77
Obrázek 5.1: Indexování jednotlivých znaků textového řetězce – stringu 90
Obrázek 8.1: Syntaktický diagram příkazu `assert` 124
Obrázek 9.1: Okno světa želvy po provedení zadaných příkazů 135
Obrázek 9.2: Syntaktický diagram možných verzí příkazu `import` 141
Obrázek 12.1: Syntaktický diagram hlavičky funkce 179
Obrázek 13.1: Význam nastavení **Show Code Context** v prostředí IDLE 189
Obrázek 14.1: Postup zpracování jednoduchého podmíněného příkazu 204
Obrázek 14.2: Vývojový diagram úplného podmíněného příkazu 205
Obrázek 14.3: Vývojový diagram zobrazující postup vykonávání rozšířeného podmíněného příkazu 206
Obrázek 15.1: Syntaktický diagram příkazu `for` 221
Obrázek 16.1: Syntaktický diagram příkazu `try` 236
Obrázek 16.2: Syntaktický diagram příkazu vyhození výjimky 242
Obrázek 17.1: Syntaxe generátorové notace seznamu 250
Obrázek 22.1: Syntaktický diagram formátovacího stringu 315
Obrázek 22.2: Syntaktický diagram nahrazovacího pole 316
Obrázek 22.3: Syntaktický diagram nahrazovaného textu 316
Obrázek 22.4: Syntaktický diagram specifikace formátu 318
Obrázek 23.1: Indexování jednotlivých znaků textového řetězce – stringu 340
Obrázek 25.1: Diamantový problém 372
Obrázek 28.1: Architektura, pro kterou není možné odvodit MRO 415
Obrázek 31.1: Strom složek s balíčky a moduly 452
Obrázek 31.2: Stromy složek s NS-balíčky 465
Obrázek 34.1: Syntaktický diagram příkazu `with` 506
Obrázek 37.1: Reakce IDLE na zápis tečky za odkazem na objekt 544

Seznam tabulek

Tabulka 5.1: Tabulka priorit operátorů (operátory, u nichž nejsou uvedeny operandy, jsou binární, infixové) 84
Tabulka 6.1: Klíčová slova jazyka Python 94
Tabulka 23.1: Tabulka zařazení jednotlivých probraných kontejnerů 337
Tabulka 23.2: Tabulka různých způsobů vykrajování 340
Tabulka 24.1: Důležité atributy a funkce modulu `os` 347
Tabulka 24.2: Probrané funkce modulu `os.path` 348
Tabulka 24.3: Režimy otevírání souboru 354
Tabulka 34.1: Tabulka přetížitelných binárních operátorů 499
Tabulka 34.2: Tabulka přetížitelných unárních operátorů 500
Tabulka 34.3: Tabulka zbylých emulačních funkcí 500
Tabulka 34.4: Tabulka konverzních funkcí 500

Seznam odboček – podšeděných bloků

Přísné a benevolentní programovací jazyky 49
Operátory jako funkční objekty 85
Statické a dynamické typování 93
Návrh podle kontraktu 124
Fyzické a logické řádky 128
Zásobník návratových adres – ZNA 213
Hešovatelné objekty 269
Předání argumentu odkazem 299
Mělké a hluboké kopie objektů 335
Terminologická vsuvka 365
Destruktor versus finalizér 496
Slabé odkazy (weak references) 550