

Začínáme programovat v jazyku **PYTHON**

- » Nepředpokládá žádné předchozí znalosti programování
- » Výklad je postaven na vybudování jednoduché aplikace a průběžném seznamování s potřebnými konstrukcemi
- » Neomezuje se na výuku kódování v Pythonu, ale učí, jak program navrhnout a postupně vyvinout a rozchodit
- » Učí čtenáře programovat podle moderních zásad a metodik



edice
začínáme s ...

Začínáme programovat v jazyku PYTHON

RUDOLF PECINOVSKÝ

GRADA Publishing

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**.

Rudolf Pecinovský

Začínáme programovat v jazyku Python

Vydala Grada Publishing, a.s.

U Průhonu 22, Praha 7

obchod@grada.cz, www.grada.cz

tel.: +420 234 264 401

jako svou 7731. publikaci

Odpovědný redaktor: Petr Somogyi

Fotografie na obálce Depositphotos/novotnyfi

Grafická úprava a sazba Rudolf Pecinovský

Počet stran 272

První vydání, Praha 2020

Vytiskly Tiskárny Havlíčkův Brod, a. s.

Dotisk 2021

© Grada Publishing, a.s., 2020

Cover Design © Grada Publishing, a. s., 2020

Cover Photo © Depositphotos/novotnyfi

Názvy produktů, firem apod. použité v knize mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

ISBN 978-80-271-1828-1 (ePub)

ISBN 978-80-271-1827-4 (pdf)

ISBN 978-80-271-1237-1 (print)

Všem, kteří se chtějí něco naučit

Stručný obsah

Stručný obsah	6
Podrobný obsah	8
Úvod	18
Část A Základy	25
1 Předehra	26
2 Superzáklady	39
3 Začínáme programovat	54
4 Základy OOP	65
5 Moduly a práce s nimi	78
Část B Připravujeme aplikaci	89
6 Základy objektové architektury	90
7 Návrh základní architektury	101
8 Kontejnery a práce s nimi	116
9 Připravujeme test	128
10 Rozhodování	139
11 Definice testu hry	152
Část C Budujeme aplikaci	163
12 Definujeme start hry	164
13 Dědění	171
14 Vytváříme svět hry	181
15 Příkazy Vezmi a Polož	189
16 Rozběhnutí aplikace	198
17 Co nám ještě chybí	205
18 Batoh a nápověda	216
19 Spustitelná aplikace	223

Část D Vylepšujeme aplikaci	235
20 Převod literálů na konstanty	236
21 Primitivní GUI.....	241
22 Kudy dál	251
Část E Přílohy	255
A Konfigurace ve Windows	256
B Použité funkce ze standardní knihovny.....	259
C Konvence pro psaní programů v Pythonu.....	263
Literatura.....	266
Rejstřík	267

Podrobný obsah

Stručný obsah	6
Podrobný obsah	8
Úvod	18
Komu je kniha určena	18
Koncepte výkladu a jeho uspořádání	19
První část	19
Druhá část	20
Třetí část	20
Čtvrtá část	20
Přílohy	20
Jazyk identifikátorů	21
Potřebné vybavení	21
Doprovodné programy	21
Použité typografické konvence	22
Odbočka – podšeděný blok	23
Zpětná vazba	24

Část A Základy 25

1 Předehra	26
1.1 Hardware a software	26
První počítače	26
Co je to program	27
Změny přístupu k tvorbě programů	27
1.2 Překladače, interprety, platformy	28
Operační systém	28
Platforma	29
Programovací jazyky	29
Překládaný program	29
Interpretovaný program	30
Porovnání	30
Hybridně zpracovávaný program	30
Jazyk versus způsob zpracování	31
1.3 Platforma Python	31
Skripty	31
Dokumentace	32
1.4 Vývojové prostředí	32
1.5 Prostředí IDLE	34
Spuštění	34
Základní popis	34
Příkazové okno	36
Restart interaktivního systému	36
Uložení záznamu seance	36

Editační okno.....	37
Umístění editovaných souborů	37
Barevné zvýraznění textu	38
Použité písmo.....	38
1.6 Shrnutí	38
2 Superzáklady	39
2.1 Počáteční mezery	39
Komentáře	39
2.2 Celá čísla	40
2.3 Reálná čísla	41
2.4 Textové řetězce – stringy.....	41
Znak # ve stringu	42
Víceřádkové stringy	42
Escape sekvence	44
Bílé znaky.....	44
2.5 Proměnné	44
Identifikátor.....	45
Jazyk identifikátorů	45
Definice a použití proměnné, přiřazovací příkaz	45
Nebezpečné změny hodnot.....	46
2.6 Literály	47
2.7 Volání funkcí	48
Příklady funkcí	48
Parametr versus argument	49
2.8 Hodnota None.....	49
Podrobnosti o volání funkcí.....	50
2.9 Zadání údajů z klávesnice.....	50
2.10 Implicitní proměnná _.....	51
2.11 Základní aritmetické operace	51
2.12 Formátovací stringy – f-stringy	52
2.13 Více příkazů na řádku	52
2.14 Shrnutí	53
3 Začínáme programovat.....	54
3.1 Definice funkce	54
Funkce je objekt, na nějž odkazuje proměnná.....	55
Dokumentační komentář	56
Získání nápovědy – dokumentace.....	56
Definice funkce je obyčejný složený příkaz	56
3.2 Definice vlastní funkce.....	56
Lokální proměnné	57
3.3 Problémy s odsazováním v IDLE	57
Sloučení více řádků do jednoho.....	58
3.4 Funkce s návratovou hodnotou	58
Shoda názvu proměnných	58
3.5 Funkce s parametry.....	59
Zadávání argumentů.....	60
Implicitní hodnoty argumentů	60
Povinně poziční a povinně pojmenované argumenty.....	61
3.6 Funkce print() a její parametry.....	61
3.7 Výrazy versus příkazy	62
3.8 Definice prázdné funkce	62
3.9 Datový typ	63
3.10 Anotace.....	63
3.11 Shrnutí	64

4 Základy OOP	65
4.1 Proč se učit objektové paradigma	65
4.2 Základní princip OOP	66
Zprávy x metody	67
Metody x funkce	67
4.3 Objekty a jejich atributy	67
Práce s objekty – kvalifikace	68
4.4 Třídy a jejich instance	69
Třída	69
Instance	70
Vytváření instancí – konstruktor, alokátor, initor	70
4.5 Definice třídy a jejich atributů	70
Dokumentační komentář	71
Příkazy těla třídy se provádějí	71
Atributy třídy	71
Dekorátory	72
Metody instancí	73
Initor	73
Speciální metody	73
Definice třídy je obyčejný příkaz	74
4.6 Práce s vytvořenou třídou a jejími instancemi	74
4.7 Použití initoru s parametry	75
Výraz na více řádcích	76
4.8 Definice prázdné třídy	76
4.9 Typy hodnot	76
4.10 Shrnutí	77
5 Moduly a práce s nimi	78
5.1 Moduly – základní informace	78
Vše je součástí nějakého modulu	78
Dva názvy objektů	79
Zdrojový soubor	79
Přeložený soubor	79
5.2 Příkaz import	80
Čistý import jiného modulu	80
5.3 Import modulu pod jiným názvem	81
Přímý import vyjmenovaných objektů	82
5.4 Vytvoření vlastního modulu	83
Název modulu	84
Kódová stránka	84
Dokumentační komentář	84
Zadané příkazy	85
5.5 Práce s vytvořeným modulem	86
Proměnná s odkazem na objekt modulu	87
Oprava načteného modulu	87
5.6 Opětovné načtení opraveného modulu	87
5.7 Shrnutí	88

Část B Připravujeme aplikaci 89

6 Základy objektové architektury	90
6.1 Předmluva	90
6.2 Architektura	91
6.3 Hlavní zásady návrhu	91
Připravenost na změny	91
CRIDP – maximální přehlednost	92

KISS – maximální jednoduchost	92
YAGNI – žádné zbytečnosti.....	92
DRY – bez kopií	93
SoC – jediný zodpovědný	93
SRP – jediná zodpovědnost.....	93
6.4 Návrhové vzory	94
6.5 Antivzory	95
6.6 Rozhraní versus implementace.....	96
PINI	96
Nezveřejňované atributy	96
6.7 Návrh programu	97
Účastníci	97
Schopnosti	97
Vlastnosti.....	97
Kódování	97
6.8 Druhy vytvářených objektů.....	98
6.9 Dva způsoby návrhu.....	98
Návrh shora dolů	98
Návrh zdola nahoru	99
Porovnání.....	99
6.10 UML diagramy	100
6.11 Shrnutí	100
7 Návrh základní architektury.....	101
7.1 Koncepce vyvíjené aplikace	101
Co to je <i>h-objekt</i>	103
7.2 Zadání	103
7.3 Účastníci – objekty vystupující ve hře.....	105
Aplikace, Hra – game	105
Svět – world	105
Prostor – place	105
Název – name	105
Příkaz – Akce – action	105
Přechod.....	106
H-objekt – item	106
Hráč.....	106
Batož – Bag – BAG	107
Úkol, cíl	107
Množství, kapacita	107
Ukončení, spuštění	107
Nápověda, přehled.....	107
7.4 Správci skupin objektů	108
Správci v naší aplikaci.....	108
7.5 Vytvoření zárodku budoucí aplikace	109
7.6 Balíčky	110
Trocha teorie	111
Název modulu.....	111
Initor balíčku.....	111
Šablona initoru balíčku	111
Rozdělení doprovodných programů do balíčků	112
Relativní import.....	113
7.7 UML diagram	114
7.8 Shrnutí	115

8 Kontejnery a práce s nimi	116
8.1 Kontejnery	116
8.2 Proměnné a neměnné objekty	116
Zvláštnosti programových kontejnerů	117
8.3 Druhy kontejnerů	117
8.4 Vytváření kontejnerů	118
Seznam – list	118
N-tice – tuple	119
Množiny – set, frozenset	120
Slovník – dict	121
8.5 Získání prvku z kontejneru	122
8.6 Projití celého kontejneru – cyklus for	122
8.7 Funkce s proměnným počtem argumentů	123
Hvězdičkový parametr	124
Hvězdičkový argument	124
Dvouhvězdičkový parametr	125
Dvouhvězdičkový argument	125
8.8 Specifika slovníků	126
items()	126
keys()	126
values()	126
8.9 Jmenné prostory	127
8.10 Shrnutí	127
9 Připravujeme test	128
9.1 Metody <code>__repr__()</code> a <code>__str__()</code>	128
9.2 Jak testovat	129
Programování řízené testy	129
Jednotkové, integrační a regresní testy	130
Možnosti testování naší hry	130
9.3 Scénáře	131
Modul <code>scenarios</code>	132
9.4 Kroky definující stav hry	132
9.5 Definice třídy <code>Step</code>	133
Anotace deklarující prvky kontejnerů	133
9.6 Definice šťastného scénáře	134
9.7 Simulace běhu hry	135
Jednoduchá simulace	136
Podrobnější simulace	137
9.8 Nezveřejňované atributy	138
9.9 Shrnutí	138
10 Rozhodování	139
10.1 Logické hodnoty	139
10.2 Terminologie výrazů	140
Operace	140
Operátor	140
Operand	140
Arita operátorů	140
Priorita operátorů	141
10.3 Porovnávání hodnot	141
Porovnání reálných čísel	141
Zřetěžené porovnávání	142
Porovnávání textů	142
Porovnávání totožnosti objektů	142

10.4	Podmíněný výraz	143
10.5	Podmíněný příkaz	143
	Jednoduchý podmíněný příkaz	144
	Větev <code>else</code>	144
	Rozhodování s více větvemi: rozšířený podmíněný příkaz	145
10.6	Tři druhy chyb	146
	Syntaktické chyby	146
	Běhové chyby	147
	Logické chyby	147
10.7	Reakce na vznik běhových chyb	147
10.8	Zachycení a ošetření výjimky	148
	Průchod programu bloky <code>try ... except ... finally</code>	148
10.9	Použití nelokálních proměnných	150
	Příkaz <code>global</code>	150
	Příkaz <code>nonlocal</code>	151
10.10	Shrnutí	151
11	Definice testu hry	152
11.1	Přířazovací výraz	152
11.2	Balíček <code>game_v1b</code>	153
11.3	Jak budeme testovat	153
	Zadání příkazu hry	153
	Odpověď a pozice	154
	Sousedé	154
	H-objekty v prostoru	154
	H-objekty v batohu	156
	Oznámení o navštíveném prostoru	156
	Souhrn	156
11.4	Vlastní test hry	156
	Úvodní testy	157
	Funkce <code>_error()</code>	158
	Funkce <code>compare_containers()</code>	158
	Proč na malá písmena	159
11.5	Spouštíme test	159
11.6	Další postup	160
11.7	Shrnutí	160

Část C Budujeme aplikaci

163

12	Definujeme start hry	164
12.1	Balíček <code>game_v1c</code>	164
12.2	Tři druhy objektů	164
12.3	Delegování zodpovědnosti	165
12.4	Funkce <code>execute_command()</code> v modulu <code>actions</code>	166
	Definice má být krátká	167
12.5	Funkce <code>_execute_empty_command()</code>	167
	Důvod použití příkazu <code>global</code>	167
12.6	Funkce <code>_execute_standard_command()</code>	168
12.7	Spuštění testu	169
12.8	Shrnutí	170
13	Dědění	171
13.1	Základní terminologie	171
	Hierarchie dědění	172

13.2	Tři druhy dědění.....	172
	Přirozené (nativní) dědění.....	172
	Dědění rozhraní	173
	Dědění implementace.....	173
13.3	LSP – substituční princip Liskové	174
13.4	Virtuální metody a jejich přebíjení	174
	Polymorfismus.....	175
13.5	Rodičovský podobjekt	175
13.6	Initory v procesu dědění	175
13.7	Definice rodičovské a dceřiné třídy	176
13.8	Násobné dědění a diamantový problém.....	177
	Návrh třídy s více bezprostředními rodiči.....	178
13.9	Zobecňování.....	179
13.10	Abstraktní třídy	180
13.11	Shrnutí	180
14	Vytváříme svět hry	181
14.1	Pravidla pro kreslení UML diagramů	181
14.2	Aktuální UML diagram.....	182
14.3	Přípravné akce, balíček game_v1d	183
14.4	Pojmenované objekty.....	183
14.5	Úprava definice třídy Item.....	184
14.6	Úprava definice třídy Place.....	185
	Vytváření slovníků pomocí metody fromkeys()	186
14.7	Vytvoření prostorů hry	187
14.8	Inicializace aktuálního prostoru a test.....	187
14.9	Shrnutí	188
15	Příkazy Vezmi a Polož.....	189
15.1	Balíček game_v1e	189
15.2	Obecná akce.....	189
15.3	Společný rodič batohu a prostorů	190
	Initor.....	190
	Inicializace.....	192
	Přidání položky	192
	Odebrání položky	192
15.4	Nebezpečí degenerovaných objektů.....	193
15.5	Úprava initoru třídy ANamed	193
15.6	Upravené definice prostorů a batohu	193
15.7	Definice akce Vezmi	194
15.8	Definice akce Polož	195
15.9	Spuštění testu	195
15.10	Shrnutí	197
16	Rozběhnutí aplikace.....	198
16.1	Balíček game_v1f	198
16.2	Definice třídy _GoTo	198
16.3	Upravujeme zprávu o chybě.....	199
	Definice funkce current_state() v modulu game.....	200
	Nová definice funkce _error() v modulu scenarios	200
	Úprava testovací funkce	201
	Úprava ošetření vyhozené výjimky.....	201
16.4	Nový test.....	201

16.5	Inicializace sousedů	202
16.6	Akce Konec	203
16.7	Shrnutí	204
17	Co nám ještě chybí.....	205
17.1	Nesplněné body zadání.....	205
	Nový balíček game_v1g	206
	Nový scénář	206
17.2	Třída Scenario	206
17.3	Chybový scénář	207
	Společný startovní krok	207
	Co vše se má zkontrolovat.....	207
	Nekorektní spuštění.....	209
17.4	Dodatečné definice testů	209
17.5	Opakované spuštění.....	210
	Opakované spuštění.....	211
	Oprava inicializace.....	211
17.6	Příkazy break a continue	212
	Příkaz break	212
	Příkaz continue.....	212
17.7	Nekorektní spuštění	213
	Test ukončení hry	213
17.8	Nezadané argumenty.....	214
17.9	Shrnutí	215
18	Batoh a nápověda	216
18.1	Vykrajování (slicing).....	216
18.2	Nový balíček game_v1h	217
18.3	Nezvednutelné h-objekty	217
	Předpona může mít širší význam	218
	Úprava metody _Take.execute()	219
18.4	Konečná kapacita batohu	219
	Metoda try_add().....	219
	Konečná verze metody _Take.execute()	220
	Ověřovací test	220
18.5	Nápověda.....	220
	Výsledek testu	221
18.6	Úprava testovací funkce	221
18.7	Rozšíření výstupu	221
18.8	Shrnutí	222
19	Spustitelná aplikace	223
19.1	Příkaz while – cyklus.....	223
19.2	Nekonečný cyklus	224
19.3	Logické operátory a operace.....	225
19.4	Balíček game_v1i	226
19.5	Jednoduché textové uživatelské rozhraní.....	226
19.6	Možnost opakovaného spuštění	227
19.7	Kontrolní tisky.....	228
	Konstanta __debug__	228
	Alternativní postup	229
	Dokonalejší postup	229

19.8	Přímé spuštění zadaného skriptu	229
	Rozpoznání režimu, v němž byl modul spuštěn.....	229
	Demonstrace.....	230
19.9	Vytvoření spustitelné aplikace	231
	Soubor typu pyz	232
19.10	Argumenty příkazového řádku	233
	Doplnění modulu game.....	233
19.11	Shrnutí	234

Část D Vylepšujeme aplikaci 235

20	Převod literálů na konstanty	236
20.1	Magické hodnoty	236
20.2	Modul textových konstant	237
20.3	Konstanty související s prostory	237
20.4	Konstanty související s h-objekty	239
20.5	Definice světa hry	239
20.6	Další úpravy	239
20.7	Shrnutí	240
21	Primitivní GUI.....	241
21.1	Balíček game_v2b	241
21.2	Změna architektury.....	241
	Třída Console	242
	Atribut io.....	243
	Úprava funkcí run() a multirun()	243
21.3	Knihovna tkinter.....	243
	Návrhový vzor Fasáda	244
21.4	Modalita dialogových oken.....	245
21.5	Primitivní dialogová okna	245
	Parametr **options	246
	Modul tkinter.messagebox.....	246
	Parametr **options	246
	Modul tkinter.simpledialog.....	247
21.6	Rodičovské okno.....	248
	Schování okna	248
21.7	Modul dialogových oken.....	248
21.8	Přímé spuštění aplikace	248
21.9	Shrnutí	250
22	Kudy dál	251
22.1	Další vylepšování.....	251
22.2	Přehled námětů.....	251
	Převod pod kvalitní grafické uživatelské rozhraní	252
	Změna světa hry	252
	Zdokonalení h-objektů	252
	H-objekty – prostory.....	252
	Rozšiřování sady příkazů	253
	Rozhovor	253
22.3	Tipy pro učitele	253
22.4	Další ukázkové příklady	254
22.5	Další zdroje.....	254

Část E Přílohy	255
A Konfigurace ve Windows	256
A.1 Definice substituovaných disků	256
A.2 Nastavování zástupce spouštějícího IDLE	257
B Použité funkce ze standardní knihovny	259
B.1 Zabudované funkce	259
abs(x)	259
bool(x)	259
dict(x)	259
eval(výraz[, globals[, locals]])	259
help(objekt)	259
input(výzva)	260
len(x)	260
list(x)	260
range(stop) range(start, stop[, step])	260
print(argumenty)	260
reload(modul); přesněji importlib.reload(modul)	260
set(x)	260
str(object)	260
super(type[, object-or-type])	260
tuple(x)	261
type(object)	261
B.2 Speciální metody	261
__init__()	261
__repr__()	261
__str__()	261
B.3 Metody třídy dir	261
fromkeys(iterable[, value])	261
B.4 Metody posloupností – Sequence	261
index(x [, i [, j]])	261
B.5 Metody třídy list	262
append(x)	262
sort(*, key=None, reverse=False)	262
B.6 Metody třídy str	262
lower()	262
split(sep=None, maxsplit=-1)	262
upper()	262
strip([chars])	262
C Konvence pro psaní programů v Pythonu	263
Uspořádání kódu	263
Jmenné konvence	264
Dokumentační komentáře (PEP 257)	265
Literatura	266
Rejstřík	267

Úvod

Python je moderní programovací jazyk, který umožňuje velmi jednoduše navrhovat jednoduché programy, ale na druhou stranu nabízí dostatečně mocné prostředky k tomu, abyste mohli s přiměřeným úsilím navrhovat i programy poměrně rozsáhlé. Je pro něj vyvinuto obrovské množství knihoven a frameworků, které uživatelům umožňují soustředit se na řešení úkol a nerozptylovat se vývojem nejružnějších pomocných podprogramů.

Popularita jazyka *Python* nepřetržitě roste. Postupně se stává klíčovým jazykem v řadě oblastí, především v těch, které souvisejí s výukou a výzkumem. Je to nejčastěji vyučovaný první jazyk na univerzitách i středních školách, je nejpoužívanějším jazykem ve statistice, programování umělé inteligence, v aplikacích využívajících strojové učení, je hlavním jazykem v oblasti analýzy dat a postupně proniká do dalších oblastí tvorby softwaru. Hraje důležitou roli i v oblasti webového programování a různých vědeckých výpočtů. Často se k němu obracejí odborníci, kteří potřebují na počítači vyřešit nějaký problém a jiné jazyky jim připadají buď příliš těžkopádné, anebo pro ně neexistují potřebné knihovny.

Python je v současné době nejlepším jazykem pro ty, kteří se nechtějí živit jako programátoři, ale jejich profese či zájem je nutí jednou za čas něco naprogramovat. Potřebují proto jazyk, který se mohou rychle naučit a v němž budou moci rychle vytvářet jednoduché programy řešící (nebo pomáhající řešit) jejich problém. Na druhou stranu ale sílí i jeho využití profesionálními vývojáři pro rozsáhlé podnikové a webové aplikace.

Komu je kniha určena

Tato kniha je určena především těm, kteří ještě nikdy neprogramovali, anebo se je to sice někdo snažil naučit, ale oni už vše zase zapomněli. Nepředpokládá žádné předběžné znalosti a dovednosti kromě základů práce s počítačem. Jejím cílem je předat čtenáři základní znalosti a naučit ho dovednosti potřebné k vytváření jednoduchých aplikací. Osvojené základy jim pak umožní, aby v případě hlubšího zájmu o programování v jazyku *Python* pokračovali některou z učebnic určených pro mírně pokročilé programátory – nejlépe samozřejmě mojí učebnicí [\[11\]](#) a doplňkovou příručkou [\[12\]](#).

Zkušenost však ukázala, že v této učebnici najdou řadu cenných informací i programátoři, kteří již mají jisté zkušenosti, ale kurzy, jimiž doposud prošli, se soustředily především na odpověď na otázku **jak**, a oni by se nyní rádi dozvěděli také odpověď na otázku **proč**.

Kniha je učebnicí programování. Učí své čtenáře navrhovat programy a dále je vylepšovat. Není učebnicí jazyka *Python* (tou je učebnice [\[11\]](#)), a proto se nesnaží probrat všechny jeho konstrukce, ale omezuje se při výkladu pouze na ty, jejichž zvládnutí je pro návrh jednoduché aplikace nezbytné.

Vedle konstrukcí jazyka ale učí čtenáře také řadu zásad moderního programování, jejichž zvládnutí je nutnou podmínkou pro všechny, kdo nechtějí zůstat u malých žákovských programů, ale chtějí se naučit efektivně vyvíjet robustní středně rozsáhlé aplikace, jejichž údržba nebude vést uživatele k chrlení nepublikovatelných výroků na adresu autora.



Dopředu se omlouvám, že se obě knihy částečně překrývají, ale cítil jsem potřebu některé věci vysvětlit jak začátečníkům, pro něž je určena tato učebnice, tak pokročilejším čtenářům, pro něž jsem psal knihu [\[11\]](#). Nepočítal jsem to, ale odhaduji, že asi 30 stránek mají obě knihy společných.

V případných příštích vydáních se pokusím tento překryv zmenšit, ale tentokrát jsem považoval za důležitější rychlost. Zájem o začátečnickou učebnici, který vyvolalo vydání příručky [\[10\]](#), mne i nakladatele zaskočil, že jsme se rozhodli upřednostnit rychlost vydání před případnou dokonalostí obsahu.

Koncepce výkladu a jeho uspořádání

Kniha je koncipována tak, aby mohla sloužit jako středoškolská učebnice programování i jako učebnice pro samouky zajímaví se o programování. Probírá vše potřebné od naprostých základů až po některé rysy, které se v začátečnických příručkách běžně neprobírají, ale jejichž znalost považuji za velmi užitečnou, protože pomáhá rychleji odhalit příčiny chyb. Kniha je rozdělena do čtyř částí.

První část

První část probírá naprosté základy, bez jejichž znalosti nelze vytvořit ani velice jednoduchou aplikaci. Naučíte se v ní pracovat s čísly a texty a dozvíte se, jak vytvářet vlastní funkce. Pak představí základy objektově orientovaného programování, na němž stojí celý *Python* a které nevědomky používají i ti, kteří tvrdí, že objektově nepracují. V závěru vás pak naučí ukládat vytvořené části programu a v případě potřeby je opět spouštět.

Doprovodné programy v první části se nesnaží nic řešit. Jsou to vesměs AHA-příklady, tj. příklady, jejichž jediným cílem je, aby si čtenář řekl: „Aha, takto to funguje.“

Druhá část

Ve druhé části začínáme vytvářet jednoduchou aplikaci s tím, že když narazíme na problém, který ještě neumíte řešit, tak vás seznámím s programovými konstrukcemi, jež se k řešení takovýchto problémů používají.

Zkušenost ukazuje, že studenti, kteří se nejprve naučí kódovat a soustředit se především na detail, mívají později velké problémy s návrhem architektury svých programů. Proto vás nejprve seznámím se základy návrhu architektury programu a principem návrhových vzorů.

Poté navrhne základní architekturu naší aplikace a začneme tuto architekturu realizovat. Paralelně vám přitom představím datové struktury umožňující efektivně pracovat se skupinami dat.

Seznámíte se s metodikou programování řízeného testy a v závěru této části se nejprve naučíte zabudovávat do programu rozhodování a opakování. Poté navrhne test chystané hry, který nám v další části bude ukazovat, kudy postupovat.

Třetí část

Ve třetí části postupně vytváříme plánovanou aplikaci. Vždy spustíme test, a ten nám řekne, co máme v dalším kroku opravit a rozchodit. Na konci této části budete mít rozchozenou jednoduchou aplikaci realizující textovou konverzační hru.

Jakmile bude aplikace funkční, doplníme k ní jednoduché uživatelské rozhraní a ukážeme si, jak aplikaci uložit, abychom ji pak mohli snadno distribuovat těm, kteří ji potřebují nebo po ní alespoň touží.

Čtvrtá část

Čtvrtá část je poměrně krátká a představuji v ní několik námětů, jak je možné aplikaci zdokonalit. V prvních dvou kapitolách ještě ukazují některé doposud nepoužité programátorské obraty a opravdu v nich něco naprogramujeme. Poslední kapitola této části reaguje na to, že cílem učebnice není vytvoření dokonalé aplikace, ale výuka základních znalostí a dovedností moderního programování. Náměty v ní uvedené jsou proto opravdu pouhé náměty pro vaše další experimenty.

Přílohy

Kniha obsahuje tři přílohy. V první seznamuji uživatele *Windows* s možnostmi definice substituovaných disků. Druhá obsahuje přehled metod ze standardní knihovny, které jsme v průběhu výuky použili. Třetí obsahuje výtah z oficiálních konvencí pro psaní kódu a dokumentačních komentářů na stránkách *Pythonu*.

Jazyk identifikátorů

Jak jsem řekl, doprovodné programy v první části jsou vesměs AHA-příklady vysvětlující probíranou konstrukci. V nich budu pro větší názornost používat české identifikátory. V AHA-příkladech budu češtinu používat i v dalších částech knihy.

Při návrhu aplikace ale budu používat identifikátory anglické, jak je ostatně v programování zvykem (a jak doporučují konvence). Komentáře a tištěné texty však zůstanou nadále české.

Potřebné vybavení

Pro úspěšné studium této knihy je vhodné mít instalovanou platformu *Python* ve verzi nejméně 3.9. Tu lze stáhnout na adrese <https://www.python.org/downloads/>. Pokud jste si však pořídili knihu před 5. říjnem 2020, na kdy je naplánováno vydání ostré verze, tak zaběhněte na <https://www.python.org/ftp/python/3.9.0/>, kde najdete poslední betu či kandidáta vydání (release candidate). Na její rodičovské stránce najdete odkazy na stránky pro všechny doposud vydané verze od 2.0 i rozpracované verze, které se teprve připravují.

Doprovodné programy

Text knihy je prostoupen řadou doprovodných programů. Budete-li si je chtít spustit a ověřit jejich funkci, potřebujete je nejprve stáhnout. Najdete je na stránce knihy na adrese¹ http://knihy.pecinovsky.cz/65_pythonvb.

Názvy zdrojových souborů s doprovodnými programy (moduly) začínají vždy písmenem **m** následovaným dvojmístným číslem kapitoly. Následují-li dvě podtržítka a text (např. `m01__prolog`), jedná se o soubor příkazů zadaných v průběhu kapitoly. Je-li za číslem malé písmeno následované jedním podtržítkem (např. `m01a_script`), jedná se o samostatný modul.

Většina těchto souborů se vyskytuje ve třech podobách se stejným názvem, ale odlišnou příponou. Soubory s příponou **py** jsou zdrojové soubory *Pythonu*, soubory s příponou **pyrec** obsahují záznam konverzace se systémem a soubory s příponou **pydoc** obsahují výpisy programů v knize i s uvedenými čísly řádků. Tyto soubory vám mají usnadnit sledování rozboru některých programů, abyste při něm nemuseli neustále listovat mezi rozbohem a rozebíraným výpisem.

¹ Číslicí šedesát pět v adrese se nevzrušujte, jedná se pouze o moje interní označení pořadí vytvářené knihy, protože bych v nich jinak bloudil.

Použité typografické konvence

K tomu, abyste se v textu lépe vyznali a také abyste si vykládanou látku lépe zapamatovali, používám několik prostředků pro odlišení a zvýraznění textu.

Termíny	První výskyt nějakého termínu a další texty, které chci zvýraznit, vysazuji tučně .
<i>Název</i>	Názvy firem a jejich produktů vysazuji <i>kurzivou</i> . Kurzivou vysazuji také názvy kapitol, podkapitol a oddílů, na něž v textu odkazuji.
Citace	Texty, které si můžete přečíst na displeji, např. názvy polí v dialogových oknech či názvy příkazů v nabídkách, vysazuji tučným bezpatkovým písmem .
<u>Odkaz</u>	Celá kniha je prošpikovaná křížovými odkazy na související pasáže. Není-li odkazovaný objekt (kapitola, obrázek, výpis programu, ...) na stejné stránce nebo na některé ze sousedních stránek, je pro čtenáře tištěné verze doplněn o číslo stránky, na níž se nachází. Čtenářům elektronické verze stačí, když na něj klepnou. Použitý prohlížeč by je měl na odkazovaný objekt ihned přenést.
Adresa	Názvy souborů a internetové adresy vysazuji obyčejným bezpatkovým písmem.
Program	Identifikátory a další části programů zmíněné v běžném textu vysazuji neproporcionálním písmem , které je v elektronických verzích pro zvýraznění tmavě červené.
zadání	V záznamech komunikace se systémem budou texty, které zadává uživatel, vysazeny tučně (v elektronických verzích pro zvýraznění tmavě modře).

Kromě výše zmíněných částí textu, které považuji za důležité zvýraznit nebo alespoň odlišit od okolního textu, najdete v knize ještě řadu doplňujících poznámek a vysvětlivek. Všechny budou v jednotném rámečku, jenž bude označen ikonou charakterizující druh informace, kterou vám chce poznámka či vysvětlivka předat.



Symbol jin-jang bude uvozovat poznámky, s nimiž se setkáte na počátku každé kapitoly. Zde vám vždy prozradím, co se v dané kapitole naučíte.



Otevřená schránka s dopisy označuje informace o doprovodných programech týkajících se dané kapitoly.



Píšící ruka označuje obyčejnou poznámku, která pouze doplňuje informace z hlavního proudu výkladu o nějakou zajímavost.



Ruka s hrozícím prstem upozorňuje na věci, které byste měli určitě vědět a na které byste si měli dát pozor, protože jejich zanedbání vás většinou dostane do problémů.



Usměváček vás bude upozorňovat na různé tipy, jimiž můžete vylepšit svůj program nebo zefektivnit svoji práci.



Mračoun vás naopak bude upozorňovat na různá úskalí programovacího jazyka nebo programů, s nimiž budeme pracovat, a bude vám radit, jak se těmto nástrahám vyhnout či jak to zařídit, aby vám alespoň pokud možno nevadily.



Obrázek knihy označuje poznámku týkající se používané terminologie. Tato poznámka většinou upozorňuje na další používané termíny označující stejnou skutečnost nebo na konvence, které se k probírané problematice vztahují. Seznam všech terminologických poznámek najdete v rejstříku pod heslem „terminologie“.



Brýle označují tzv. „poznámky pro šfouraly“, ve kterých se vás snažím seznámit s některými zajímavými vlastnostmi probírané konstrukce nebo upozorňuji na některé souvislosti, avšak které nejsou k pochopení látky nezbytné.

Odbočka – podšeděný blok

Občas je potřeba vysvětlit něco, co nezapadá přímo do okolního textu. V takových případech používám podšeděný blok se silnou čarou po straně. Tento podšeděný blok je takovou drobnou odbočkou od ostatního výkladu. Nadpis podšeděného bloku pak najdete i v podrobném obsahu mezi nečíslovanými nadpisy.

Zpětná vazba

Předem přiznávám, že tato kniha je sice mou šedesátou pátou učebnicí, ale současně je to moje první učebnice programování v jazyku *Python*. Nelze proto vyloučit, že přestože knihu četlo několik lektorů, mohou se v ní objevit přehlédnutí, která nemá redaktor šanci zachytit a opravit.

Pokud vám proto bude někde připadat text nepříliš srozumitelný nebo budete mít nějaký dotaz (ať už k vykládané látce či použitému vývojovému prostředí), anebo pokud v knize objevíte nějakou chybu či budete mít návrh na nějaké její vylepšení, neostýchejte se poslat na adresu rudolf@pecinovsky.cz e-mail s předmětem [65 PYTHON NZ DOTAZ](http://knihy.pecinovsky.cz/65_python39vb).

Bude-li se dotaz týkat něčeho obecnějšího nebo to bude upozornění na chybu, pokusím se co nejdříve zveřejnit na stránce knihy http://knihy.pecinovsky.cz/65_python39vb odpověď i pro ostatní čtenáře, kteří by mohli o danou chybu zakopnout, nebo by je mohl obdobný dotaz napadnout za pár dní, anebo jsou natolik ostýchaví, že si netroufnou se sami zeptat.

Část A

Základy

První část knihy vás seznámí se základními konstrukcemi jazyka *Python*, bez jejichž znalosti se při sestavování programu neobejdete. Nejprve poskytne základní informace o počítačích, programech a programovacích jazycích. Pak vás seznámí se zadáváním čísel a textů, naučí vás definovat funkce a třídy a na závěr vám ukáže, jak ukládat a načítat vaše programy.

Kapitola 1

Přehra



Co se v kapitole naučíte

Tato kapitola bude spíše teoretická. Má vás především seznámit s termíny, které pak budeme v textu knihy používat. Zároveň vám představí vývojové prostředí IDLE, které budeme v dalším textu používat k zadávání a spouštění doprovodných programů.

1.1 Hardware a software

První počítače

První samočinný počítač řízený programem byl mechanický a navrhl jej v roce 1837 Charles Babbage. Tou dobou byl také publikován první program. Napsala jej Ada Lovelace a vyšel v jejích poznámkách překladatele ke knize o Babbageově stroji.

První fungující elektromechanický samočinný počítač zprovoznil v roce 1938 Konrad Zuse v Německu. V průběhu druhé světové války vznikly další počítače v Anglii a ve Spojených státech.

Po válce vznikl poměrně rychle trh s počítači. Počítače konstruovali převážně muži a programovali je převážně ženy, protože řada mužů považovala programování za něco přízemního, co není hodno jejich intelektu. Situace se začala měnit až v šedesátých letech, protože se postupně ukazovalo, že podniky utratí mnohem více za programové vybavení než za vlastní počítač.

Navíc většina důvtipných řešení a patentů v oblasti hardwaru rychle zastarala a byla překonána mnohem důvtipnějšími řešeními někoho jiného, kdežto objevy na poli softwaru přežívaly celá desetiletí. V současné době se už proto hlavní vývoj neodehrává v oblasti hardwaru, ale softwaru.

Co je to program

Program bychom mohli charakterizovat jako v nějakém programovacím jazyku zapsaný předpis popisující, jak má procesor, pro nějž je program určen (v našem případě počítač), splnit zadanou úlohu.

Cílovým procesorem nemusí být vždy počítač. Oblíbeným příkladem programů jsou např. kuchařské předpisy. V předpočítačových dobách zaměstnávaly některé instituce (např. armáda) velké skupiny počtářů² řešících na mechanických kalkulačkách výpočty podle algoritmů, které dnes zakódujeme v nějakém programovacím jazyku a předhodíme počítači.

Na programu je důležité, že musí být napsán v nějakém programovacím jazyku, kterému rozumí programátor. Napsaný program je pak jiným programem (překladačem) převeden do „jazyka“, kterému rozumí počítač.

Vybrat jazyk pro kuchařský předpis je jednoduché, vybrat jazyk pro počítač je mnohem složitější. Vlastnosti použitého jazyka totiž naprosto zásadně ovlivňují jak rychlost vývoje programu, tak i rychlost a kvalitu výsledných programů. Proto také prošly programovací jazyky i metodiky jejich používání celou řadou revolučních změn.

Změny přístupu k tvorbě programů

Počítače byly postupně nasazovány v dalších a dalších oblastech a programátoři pro ně vytvářeli stále dokonalejší programy. Programy byly čím dál rafinovanější a složitější, a to začalo vyvolávat velké problémy. Programátoři totiž přestávali být schopni své programy rozchodit, a když je vítězně rozchodili, nedokázali z nich v rozumném čase odstranit chyby, které uživatelé v programu objevili.

Tato krize vedla k zavádění nejrůznějších metodik, které měly jediný cíl: pomoci programátorům psát spolehlivé a snadno upravovatelné programy. V padesátých letech minulého století se tak prosadily vyšší programovací jazyky, v šedesátých letech modulární programování, v sedmdesátých letech na ně navázalo strukturované programování a v průběhu osmdesátých a zejména pak devadesátých let ovládlo programátorský svět objektově orientované programování, jehož vláda pokračuje dodnes. (Jednotlivé termíny si za chvíli probereme podrobněji.)

Hlavním cílem programátorů v počátcích programování bylo, aby jejich programy spotřebovaly co nejméně paměti a byly co nejrychlejší. Tehdejší počítače totiž měly paměti málo, byly z dnešního hlediska velice pomalé a jejich strojový čas byl drahý. Se stoupající složitostí programů však byly takto psané programy stále méně stabilní a stále hůře udržovatelné. Současně s tím, jak klesala cena počítačů, jejich strojového času i paměti, začínal být nejdražším článkem v celém vývoji člověk.

² Příznějme si, že to tehdy byly většinou počtářky, protože muži dělají při práci podobného druhu příliš mnoho chyb. Takovéto „lidské počítače“ pomáhaly např. v Sovětském svazu s vývojem atomové pumy či s prvními lety do vesmíru.