

Python

snadno a rychle



Bez předchozích
znalostí programování

Rychlé zvládnutí
základů jazyka

Srozumitelný výklad
složitějších témat

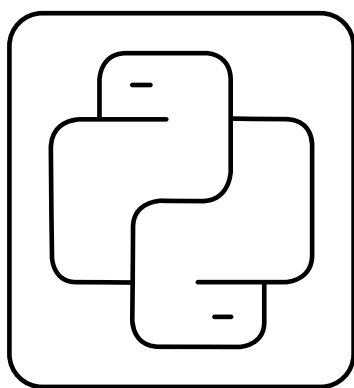
Vytvoření první
jednoduché aplikace

Rudolf Pecinovský



Python

snadno a rychle



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.

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.

Rudolf Pecinovský

Python snadno a rychle

Vydala Grada Publishing, a.s.

U Průhonu 22, Praha 7

obchod@grada.cz, www.grada.cz

tel.: +420 234 264 401

jako svou 9826. publikaci

Odpovědný redaktor Petr Somogyi

Grafická úprava a sazba Jaroslav Kolman

Počet stran 184

První vydání, Praha 2024

Vytiskly Tiskárny Havlíčkův Brod, a.s.

© Grada Publishing, a.s., 2024

Cover Design © Grada Publishing, a.s., 2024

Cover Photo © Depositphotos/iunewind

ISBN 978–80–271–7678–6 (ePub)

ISBN 978–80–271–7677–9 (pdf)

ISBN 978–80–271–5519–4 (print)

Obsah

O autorovi	13
Úvod	14
Komu je kniha určena	14
Pro koho kniha není	14
Čím se kniha liší od běžných kurzů a učebnic	15
Požadované znalosti	15
Potřebné vybavení	16
Doprovodné programy	16
Použité typografické konvence	17
Zpětná vazba	18
 1 Přehled	19
1.1 Platforma Python	19
1.1.1 Součásti standardní instalace	19
1.1.2 Instalace Pythonu	20
1.1.3 Spuštění	20
1.1.4 Skripty	21
1.1.5 Dokumentace	23
1.2 Řádkový interpret	24
1.2.1 Odsazování a pokračovací řádky	25
 2 Vývojová prostředí	27
2.1 Vývojové prostředí	27
2.2 Prostředí IDLE	28
2.2.1 Spuštění prostředí IDLE	28
2.2.2 Základní popis	29
2.2.3 Příkazové okno a jeho výzvy	30
2.2.4 Opětne zadání dříve zadaných příkazů	30
2.2.5 Restart interaktivního systému	30
2.2.6 Uložení záznamu seance	31
2.3 IDLE versus řádkový interpret	32
2.3.1 Zobrazování výpisů programů	32

3	Zadávání hodnot a volání funkcí	33
3.1	Komentáře	33
3.1.1	Zakomentování a odkomentování kódu	33
3.2	Celá čísla	34
3.3	Reálná čísla	35
3.4	Textové řetězce – stringy	36
3.4.1	Znak # ve stringu	37
3.4.2	Víceřádkové stringy	37
3.5	Funkce	38
3.5.1	Volání funkce	38
3.5.2	Návratová hodnota funkce, hodnota None	39
3.6	Znovu stringy	40
3.6.1	Bílé znaky	40
3.6.2	Escape sekvence	40
3.7	Literály	41
3.7.1	Sousedící stringové literály	42
4	Proměnné, příkazy a výrazy	43
4.1	Proměnné a přiřazovací příkazy	43
4.1.1	Proměnné	43
4.1.2	Přiřazovací příkaz	43
4.1.3	Identifikátor	44
4.1.4	Konvence pro podobu identifikátorů	45
4.2	Výrazy a příkazy	46
4.2.1	Zadání skupiny hodnot	46
4.2.2	N-tice hodnot	47
4.3	Více příkazů na řádku	48
4.4	Zjištění typu hodnoty	49
4.5	Základní aritmetické a stringové operace	50
4.5.1	Sčítání (+)	50
4.5.2	Odčítání (–)	50
4.5.3	Násobení (*)	50
4.5.4	Dělení (/)	51
4.5.5	Celá část podílu (//)	51
4.5.6	Zbytek po dělení (%)	52
4.5.7	Umocňování (**)	52
4.6	Složené přiřazovací příkazy	53
4.7	Formátovací stringy – f-stringy	54
4.7.1	Závěrečné rovnítko	55

5	Práce s objekty	56
5.1	Vše je objekt	56
5.1.1	Co je to objekt	56
5.1.2	Třída – instance – typ	57
5.1.3	Vytváření objektů	57
5.1.4	Atributy objektů a jejich kvalifikace	57
5.1.5	Textové podpisy objektů	59
5.2	Práce s objekty = práce s odkazy	60
5.2.1	Analogie: proměnná je telefon a odkaz je telefonní číslo	60
5.3	Získání nápovědy – dokumentace	61
5.3.1	Argument zadán	61
5.3.2	Nápověda k některým operátorům a konstrukcím jazyka	63
5.3.3	Bez argumentu	63
6	Moduly a balíčky	66
6.1	Moduly – základní informace	66
6.1.1	Vše je součástí nějakého modulu	66
6.1.2	Dva názvy objektů	67
6.1.3	Zdrojový soubor	67
6.1.4	Přeložený soubor	68
6.2	Příkaz <code>import</code>	68
6.2.1	Import je jen jiný druh přiřazení	68
6.2.2	Čistý import jiného modulu	69
6.2.3	Import modulu pod jiným názvem	70
6.2.4	Opakovaný import nic nenačítá	71
6.3	Přímý import atributů	72
6.3.1	Přímý import více atributů současně	73
6.3.2	Hromadný přímý import všech atributů daného modulu	74
6.4	Vytvoření vlastního modulu	75
6.5	Demonstrační modul <code>m06a_module_demo</code>	75
6.5.1	Název modulu	75
6.5.2	Zdrojový kód modulu <code>m06a_module_demo</code>	76
6.5.3	Kódová stránka	77
6.5.4	Dokumentační komentář	77
6.5.5	Import ladicího modulu a kontrolní tisky	77
6.5.6	Zadané příkazy	78

6.6	Práce s vytvořeným modulem	78
6.6.1	Proměnná s odkazem na objekt modulu	79
6.7	Balíčky	79
6.7.1	Trocha terminologie	79
6.7.2	Název modulu – balíčku	80
6.7.3	Initor balíčku	80
6.8	Umístění modulů s doprovodnými programy	80
6.9	Co dělat, když interpret na modul nevidí	81
6.9.1	Detekce a změna pracovní složky	82
7	Definice funkcí	84
7.1	Možné podoby	84
7.1.1	Dokumentační komentář	86
7.1.2	Definice prázdné funkce	86
7.1.3	Funkce je objekt, na nějž odkazuje proměnná	87
7.2	Lokální proměnné	88
7.2.1	Funkci můžeme definovat její vlastní atributy	89
7.3	Funkce s návratovou hodnotou	90
7.4	Funkce s parametry	91
7.4.1	Zadávaní argumentů	92
7.4.2	Inicializované parametry a implicitní hodnoty argumentů	92
7.5	Současné vrácení více hodnot	93
7.6	Příkaz global	94
7.7	Funkce print() a její parametry	94
8	Robot Karel a příkaz with	96
8.1	Knihovny	96
8.1.1	Knihovna Karelcz76	97
8.2	Signatura funkcí	98
8.3	Robot Karel a jeho svět	98
8.3.1	Vytvoření světa	99
8.3.2	Vytvoření robota	100
8.3.3	Výchozí sada akcí	101
8.3.4	Testy	104
8.4	Příkaz with	105
8.5	Ukončení práce s daným světem robotů	106
8.6	Modul util.for_karel	106

9	Rozhodování	108
9.1	Kontejnery	108
9.2	Logické hodnoty	109
9.3	Terminologie výrazů	109
9.4	Porovnávání hodnot	110
9.4.1	Porovnání reálných čísel	110
9.4.2	Zřetěžené porovnávání	111
9.4.3	Porovnávání textů	111
9.4.4	Porovnání hodnot a totožnost objektů	111
9.5	Podmíněný příkaz	112
9.5.1	Jednoduchý podmíněný příkaz	112
9.5.2	Vnořování složených příkazů	115
9.5.3	Větev else – úplný podmíněný příkaz	116
9.5.4	Rozhodování s více větvemi: rozšířený podmíněný příkaz	118
9.6	Přepínač – příkaz match ... case	119
9.7	Logické operátory a operace	120
9.8	Podmíněný výraz	122
10	Opakování kódu, cykly	123
10.1	Předehra	123
10.2	Příkaz while – cyklus se vstupní podmínkou	124
10.2.1	Zanořování cyklů	125
10.3	Nekonečný cyklus	126
10.4	Příkaz break – cyklus s podmínkou uprostřed	127
10.5	Cyklus s koncovou podmínkou	128
10.6	Zdroje hodnot	128
10.6.1	Rozbalovací hvězdička	129
10.6.2	Stringy	129
10.6.3	Objekt typu range	129
10.6.4	Objekt typu enumerate	130
10.7	Příkaz for – cyklus s parametrem	130
10.8	Příkaz continue	132

11	Kontejnery	133
11.1	Kontejnery	133
11.2	Nejprve trochu terminologie	134
11.2.1	Proměnné a neměnné objekty	134
11.2.2	Hešovatelné objekty	135
11.3	Druhy kontejnerů	136
11.4	Vytváření kontejnerů	137
11.4.1	Vytváření kontejnerů prostřednictvím literálů	137
11.4.2	Vytváření kontejnerů prostřednictvím konstruktorů	138
11.4.3	Vytváření kontejnerů prostřednictvím generátorové notace	141
11.4.4	Generátory lze použít jen jednou	143
12	Práce s kontejnery	144
12.1	Vytvoření výchozí sady pomocných kontejnerů	144
12.2	Test přítomnosti prvku v kontejneru	145
12.3	Získání a nastavení prvku posloupnosti a slovníku	146
12.3.1	Zárodek hry kámen, nůžky, papír	147
12.4	Procházení kontejnerů – cyklus for	148
12.4.1	Procházení slovníků – pohledy	149
12.4.2	Konvence pro názvy slovníků	151
12.5	Kopírování objektů	151
12.6	Vykrajování (slicing) posloupností	151
12.7	Přidání a odebrání prvku kontejneru	153
12.8	Funkce s proměnným počtem argumentů	154
13	Ošetřování chyb	155
13.1	Tři druhy chyb	155
13.1.1	Syntaktické chyby	155
13.1.2	Běhové chyby	156
13.1.3	Logické chyby	156
13.2	Reakce na vznik běhových chyb	156
13.3	Zachycení a ošetření výjimky	157
13.3.1	Průchod programu bloky try ... except ... finally	158

13.4	Demonstrační modul <code>m13a_demo_výjimky</code>	158
13.5	Chování programu za běhu	160
13.6	Analýza chybové zprávy	162
13.7	Ladění a kontrolní tisky – modul <code>dbg</code>	162
13.7.1	Základní funkce modulu <code>dbg</code>	162
14	Vytváříme aplikace	165
14.1	Spuštění versus import zadaného skriptu	165
14.1.1	Rozpoznání režimu, v němž byl modul zaveden	166
14.1.2	Modul <code>m14a_script</code>	166
14.2	Argumenty příkazového řádku	167
14.3	Rozsáhlejší aplikace	169
14.3.1	Umístění	170
14.3.2	Modul <code>io</code> pro ošetření vstupu	170
14.3.3	Modul <code>game</code> s jádrem hry	171
14.3.4	Řídicí modul <code>main</code>	173
14.4	Vytvoření spustitelné aplikace	173
	Závěr	175
	Literatura	177
	Rejstřík	179



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

O autorovi

Ing. Rudolf Pecinovský, CSc. je absolventem *Fakulty elektrotechnické ČVUT* z roku 1979. Titul CSc. získal v Ústavu teorie informace a automatizace ČSAV v roce 1983. Od počátku 80. let učí a publikuje, přičemž svůj výzkum soustředí především na oblast vstupních kurzů moderního programování pro naprosté začátečníky. V současné době učí na *Fakultě jaderné a fyzikálně inženýrské ČVUT* a na *Fakultě informatiky a statistiky Vysoké školy ekonomické v Praze*. Vedle toho vyučuje ještě v řadě kurzů pro začátečníky i profesionální programátory. Doposud mu vyšlo přes 60 knih, které byly přeloženy do pěti jazyků. Většina jeho knih je zaměřena na výuku moderního programování a na umění návrhu objektově orientované architektury.



Úvod

Python je moderní programovací jazyk, který umožňuje velmi jednoduše navrhovat jednoduché programy, ale na druhou stranu nabízí dostatečně účinné 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 nejrůznějších pomocných podprogramů.

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 jednoduchého naprogramovat. Potřebují proto jazyk, který se mohou rychle naučit a v němž budou moci rychle vytvářet jednoduché programy řešící (nebo pomáhající řešit) jejich problém.

Komu je kniha určena

Tato kniha je určena pro ty, kteří se chtějí co nejrychleji naučit základy jazyka, aby mohli co nejdříve začít vytvářet své jednoduché skripty.

Nezáleží na tom, jestli ještě nikdy neprogramovali, anebo se je to sice někdo snažil naučit, ale oni už většinu látky zapomněli. Kniha 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 skriptů.

Pro koho kniha není

Kniha není určena pro ty, kteří chtějí poznat jazyk *Python* hlouběji a naučit se v něm vytvářet nějaké netriviální aplikace. Jejich tvorba vyžaduje poněkud hlubší znalost programování, než jakou nabízí tato kniha, a zejména pak znalost objektového programování. Pro tyto náročnější zájemce je určena učebnice *Začínáme programovat v jazyku Python – Třetí, aktualizované a doplněné vydání* [14].



Dopředu se omlouvám, že se tato kniha částečně překrývá se zmiňovanou učebnicí [14]. Některé základní věci je prostě třeba vysvětlit všem začátečníkům. Kniha [14] probírá konstrukce jazyka podrobněji, přidává výklad základních pravidel programování rozsáhlejších systémů a hlavně probírá řadu témat, která je důležité znát a ovládat při tvorbě netriviálních aplikací.

Čím se kniha liší od běžných kurzů a učebnic

Kniha neprobírá tvorbu tříd a její pravidla. Jejich výklad sice najdete na konci většiny začátečnických kurzů, ale tyto znalosti tvůrci jednoduchých programů nikdy nepoužijí a vše beztak rychle zapomenou. K jejich rozumnému použití je totiž třeba vše procvičit na nějakém netriviálním příkladu, jak je tomu např. ve zmiňované učebnici [14], ale nic takového v běžných kurzech nenajdete.

Na druhou stranu tato kniha probere nejenom základní konstrukce jazyka, ale vysvětlí i řadu jejich vedlejších efektů, které vám pomohou při odhalování chyb ve vytvořených programech. Jak napsal jeden čtenář: „*Myslel jsem si, že nejsem žádné programátorské ucho. Když jsem ale přečetl vaši učebnici, otevřel jsem oči a hubu. Konečně mi někdo vysvětlil věci, které jsem z doposud přečtených knih a absolvovaných kurzů nepochopil.*“

Požadované znalosti

Před chvílí jsem řekl, že: „*Kniha nepředpokládá žádné předběžné znalosti a dovednosti kromě základů práce s počítačem.*“ Jak jsem ale z různých reakcí zjistil, představa o tom, co to jsou základy práce s počítačem, se mezi studenty dost různí. Řada z nich se domnívá, že když umějí počítač zapnout a poklepáním spustit aplikaci, tak základy práce zvládají. Bohužel, pro studium této učebnice toho musíte umět trochu více. Budete potřebovat následující dovednosti:

- Vědět, jaký je rozdíl mezi souborem a složkou, umět vytvořit novou složku a vložit či přesunout do ní požadované soubory.
- Stáhnout soubor ze zadané webové stránky a uložit jej do zadané složky.
- Rozbalit zadaný archivní soubor ve formátu ZIP do zadané složky.
- Otevřít okno příkazového řádku (okno konzoly).
- Zadávat v okně příkazového řádku požadované příkazy.

Není to mnoho, ale bez těchto znalostí se při čtení dalšího textu neobejdete. Na druhou stranu vám chybějící znalosti jistě rychle pomůže často doplnit nápověda od umělé inteligence, která je již všudypřítomná a počet jejích nesprávných odpovědí se rychle snižuje.

Potřebné vybavení

Pro úspěšnou práci s touto knihou je vhodné mít instalovanou platformu *Python*. Optimální by byla verze 3.13, pro kterou byla kniha psána, anebo nějaká mladší. Drtivá většina příkladů bude ale bez problémů chodit i na verzích starších.

Poslední verzi *Pythonu* lze stáhnout na adrese <https://www.python.org/downloads/>. Při instalaci určitě zaškrtněte, že chcete instalovat „**tcl/tk and IDLE**“ a že chcete instalovat dokumentaci, i když se dá používat i její on-line verze na stránkách *Pythonu*.

U některých operačních systémů je instalace poněkud složitější, tak jsem se rozhodl, že budu šetřit místem a nejčastější odchylky shrnu v samostatném dokumentu na webových stránkách knihy.

Doprovodné programy

Text knihy je prostoupen řadou doprovodných programů. Budete-li je chtít spustit a ověřit jejich funkci, potřebujete je nejprve stáhnout. Najdete je na stránce knihy na adrese¹ http://knihy.pecinovsky.cz/76_python_sr. Na stránce najdete i podrobný výklad obsahu doprovodných programů a časem i pár doplňkových textů.

Mimo jiné zde budou i soubory obsahující všechny výpisy programů v knize i s uvedenými čísly řádků. Tyto soubory vám mají usnadnit sledování rozboru některých programů, abyste si je mohli vytisknout, položit vedle knihy či čtečky a nemuseli při čtení rozboru programů neustále listovat mezi rozbohem a rozebíraným výpisem.

Kromě toho tu najdete některé další doplňkové soubory.

¹ Číslem 76 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, stejně jako názvy souborů a internetové adresy.
Citace	Texty, které si můžete přečíst na displeji, např. názvy polí v dialogových oknech či názvy příkazů v nabídkách, vysazuji tučným bezpatkovým písmem .
<i>Odkaz</i>	V knize najdete křížové odkazy na související pasáže. Pro čtenáře tištěné verze je doplněný o číslo stránky, na niž se nachází. Čtenářům elektronické verze stačí, když na něj klepnou.
Program	Identifikátory a další části programů zmíněné v běžném textu vysazuji neproporcionálním písmem , které je v elektronických verzích pro zvýraznění tmavě červené.
<u>Keyword</u>	Klíčová slova jazyka jsou navíc pro zvýraznění vysazena tučně a podtržena.
'string'	Textové řetězce neboli stringy v programech jsou fialové a žlutě (v tištěné knize šedě) podbarvené.
>>>	Značná část výpisů zaznamenává konverzaci mezi uživatelem a interpretem jazyka. Výzvy interpretu k zadání příkazu jsou vysazeny s jiným podbarvením a barvou písma, aby je bylo možné snadno odlišit od zadání uživatele a odpovědi na zadaný příkaz.
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 doplňující poznámky a tipy, označené příslušnými ikonami.



Poznámka doplňuje informace z hlavního proudu výkladu o nějakou zajímavost, upozorní, na co si dávat pozor, nebo uvede souvislosti, které nejsou pro pochopení probírané problematiky nezbytné.



Tip vás upozorní na různé možnosti, jimiž můžete vylepšit svůj program nebo zefektivnit svoji práci.

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.

Zpětná vazba

Tato kniha je prvním pokusem o napsání knihy pro začátečníky, kteří netouží pronikat hlouběji do tajů programování. Nelze proto vyloučit, že přestože ji 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 `76_PYTHON_SR_DOTAZ`.

Bude-li se dotaz týkat něčeho obecnějšího nebo to bude upozornění na chybu, pokusím se zveřejnit na stránce knihy http://knihy.pecinovsky.cz/76_python_sr odpověď i pro ostatní čtenáře, kteří by mohli o danou chybu zapoknout, nebo by je mohl obdobný dotaz napadnout za pár dní, anebo jsou natolik ostýchaví, že si netroufnou se sami zeptat.

Dopředu se ale omlouvám, že na poštu často odpovídám se značným zpožděním, protože jsem se nechal přemluvit k napsání několika dalších knih a rychleji to stihnout neumím.

1

Předehra

Co se v kapitole naučíte

Začneme tím, že se naučíte instalovat prostředí *Python* spolu s jeho zabudovaným vývojovým prostředím *IDLE*.



1.1 Platforma Python

Kniha má sice v názvu, že je učebnicí programování v jazyku *Python*, ale samotný jazyk vám není k ničemu do té doby, než získáte nástroje k tomu, abyste jej mohli používat. Pojdme je probrat a následně instalovat.

1.1.1 Součásti standardní instalace

Pro další práci potřebujete mít instalovanou platformu *Python*. Její součástí je:

- **Překladač**, který váš program přeloží do bajtkódu platformy a propojí jej s použitými knihovnami.
- **Interpret**, jenž přeložený program, tj. bajtkód, příkaz za příkazem interpretuje, a tím jej provede.
- **Knihovna**, v níž jsou připravené podprogramy realizující nejčastější operace, abyste je nemuseli všechny vytvářet znovu sami.

To vše je součástí minimální instalace, kterou nabízejí některé distribuce *Linuxu*. Součástí standardní instalace však bývá více. Patří do ní:

- Poměrně podrobná **dokumentace** s definicí jazyka, popisem instalovaných knihoven, jednoduchým tutoriálem a řadou dalších užitečných informací.
- Součástí dokumentace je i **tutoriál**, jenž vás seznámí s naprostými základy.

- Knihovna **Tkinter** pro tvorbu grafického uživatelského rozhraní (GUI).
- Nad touto knihovnou vybudované jednoduché vývojové prostředí **IDLE**.
- Program **pip** sloužící k instalaci externích knihoven.

Protože standardní instalace obsahuje prakticky všechny klíčové součásti, které pro vývoj programů v *Pythonu* potřebujete, bývá označována anglickým marketingovým heslem *battery included*.

1.1.2 Instalace Pythonu

Pro instalační soubory si dojděte na stránku <https://www.python.org/downloads/>, kde zadáte, pro který operační systém chcete *Python* instalovat. Vedle obligátních systémů *Windows* a *macOS* zde najdete informace o instalaci na řadu systémů IBM od AIX po OS/390, pro Solaris a HP-UX a řadu dalších systémů.

Instalace pro systémy *Linux* jsou nabízeny ve zdrojové formě, aby se při překladu a sestavní přizpůsobily příslušné distribuci. Řada distribucí vám ale nabídne *Python* předinstalovaný, anebo je instalační program součástí dané distribuce.

1.1.3 Spuštění

Součástí standardní instalace bývá nastavení operačního systému, které zabezpečí, aby systém správně zpracoval soubory s některou z definovaných přípon začínajících **py**.

Příkaz ke spuštění *Pythonu* z příkazového řádku závisí na použitém operačním systému.

- V systému *Windows* se *Python* může spustit příkazem **py** nebo **python**.
- V systémech *Linux* a *macOS* se většinou spouští příkazem **python3**. Někde bude fungovat i příkaz **python**, ale u něj si musíte dát pozor, protože na některých systémech tímto příkazem spustíte starší *Python* 2.
- Někde je třeba uvést plnou verzi, takže chcete-li spustit např. verzi 3.13, musíte ji spustit příkazem **python3.13**.

Já budu v doprovodných programech používat příkaz **py**, protože pracuji pod *Windows*, v němž pracuje 92 % čtenářů, a tento příkaz je ze všech uvedených nejkratší.

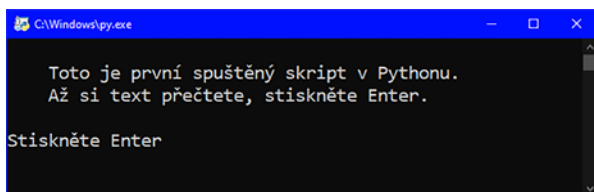
Zadáte-li uvedený příkaz bez argumentů, spustíte řádkový interpret. Jako vývojové prostředí je však řádkový interpret v konzolovém okně příliš těžkopádný a je vhodný maximálně pro spouštění jednotlivých skriptů. V podkapitole [2.2 Prostor IDLE](#) na straně [28](#) si proto představíme něco maličko dokonalejšího.

1.1.4 Skripty

Termínem skript (anglicky script) označujeme spustitelný textový soubor obsahující kód v nějakém skriptovacím jazyce. Klíčové je, že tento soubor musí být spustitelný, tj. že na počítači musí existovat program, který umí tento soubor načíst a zadaný kód provést. Operační systém by pak měl umět poznat, kterému programu má zadaný soubor předat ke zpracování.

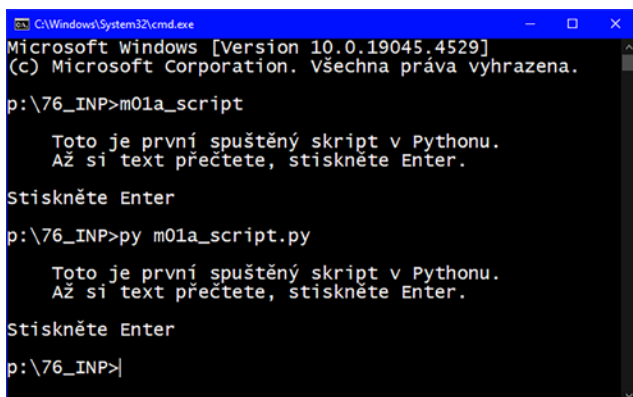
Soubory se skripty v jazyku *Python* by měly mít příponu **py**, jejich přeložené verze pak příponu **pyc**. Máte-li správně instalovaný *Python*, tak pokud takový soubor zadáte ke spuštění, měl by být spuštěn program **Python**, jenž kód v daném souboru provede.

Máte-li již instalovaný *Python* a stažené doprovodné soubory zmiňované v pasáži *Potřebné vybavení* na straně 16, můžete si vše vyzkoušet. Otevřete složku, do níž jste umísťovali doprovodné programy, najdete v ní soubor **m01a_script.py** a poklepejte na něj. Pracujete-li ve *Windows*, mělo by se otevřít okno konzoly z obrázku 1.1, pracujete-li v jiném operačním systému, měl by se otevřít jeho ekvivalent.



Obrázek 1.1: Okno otevřené po poklepání na skript **m01a_script.py**

Nechce-li se vám spouštět skript poklepáním, ale budete pracovat v konzolovém okně, přesuňte se do složky s daným skriptem a zadejte v příkazovém řádku název souboru – v našem případě **m01a_script**. Pokud se při instalaci nastavilo správně sdružení přípony souboru s programem, nemusíte zadávat ani příponu, jak je to předvedeno v prvním příkazu na obrázku 1.2.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. Všechna práva vyhrazena.

p:\76_INP>m01a_script

    Toto je první spuštěný skript v Pythonu.
    Až si text přečtete, stiskněte Enter.

Stiskněte Enter

p:\76_INP>py m01a_script.py

    Toto je první spuštěný skript v Pythonu.
    Až si text přečtete, stiskněte Enter.

Stiskněte Enter

p:\76_INP>|
```

Obrázek 1.2: Spouštění skriptu `m01a_script.py` v konzolovém okně

Pro ty, kteří nemají zkušenosti s čtením konzolového okna, opíšu jeho obsah do následujícího výpisu:

```
1 Microsoft Windows [Version 10.0.19045.4529]
2 (c) Microsoft Corporation. Všechna práva vyhrazena.
3
4 P:\76_INP>m01a_script
5
6     Toto je první spuštěný skript v Pythonu.
7     Až si text přečtete, stiskněte Enter.
8
9 Stiskněte Enter
10
11 P:\76_INP>py m01a_script.py
12
13     Toto je první spuštěný skript v Pythonu.
14     Až si text přečtete, stiskněte Enter.
15
16 Stiskněte Enter
17
18 P:\76_INP>
```

Na řádku 4 jsem zadal název skriptu bez přípony, jak jsem říkal před chvílí. Operační systém spustil příslušný skript.

Nebude-li počítač správně reagovat na pouhé zadání názvu souboru se skriptem, vyvolejte *Python* explicitně² a předejte mu název skriptu jako argu-

² Explicitně znamená přímo, zjevně, doslova. V našem případě zadáte přímo příkaz, kterým spustíte interpret *Pythonu*, tj. příkaz `py`, `python` nebo `python3`.

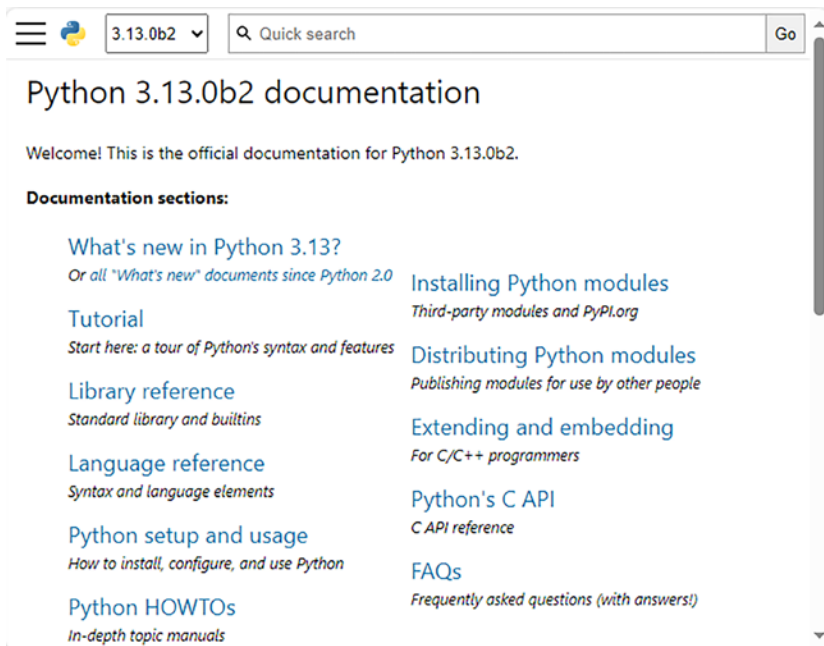
ment, jak je to předvedeno na řádce 11. (Uživatelé systémů *macOS* a *Linux* musí místo uvedeného `py` zadat `python3`.)

V takovém případě ale musíte zadat název souboru i s příponou. Na druhou stranu můžete takto předat *Pythonu* soubor s libovolnou příponou, protože *Python* se začne zabývat jeho obsahem a najde-li v souboru zdrojový kód *Pythonu*, bude spokojen.

1.1.5 Dokumentace

Říkali jsme si, že součástí standardní instalace je i poměrně podrobná dokumentace. Přítomnost dokumentace je nesmírně důležitá (a troufám si tvrdit, že je jednou z příčin úspěchu jazyka), protože *Python* přichází (ostatně jako většina současných platform) s obrovskou nabídkou možností, které si průměrný lidský mozek nedokáže zapamatovat. Proto jistě oceníte možnost rychlého získání potřebných informací jak o samotném jazyku, tak o instalovaných knihovnách. Navíc zde (tj. v doprovodné dokumentaci) najdete i několik rad a doporučení.

Dokumentace se distribuuje jako složka se sadou souborů ve formátu HTML. Uživatelé *Windows* najdou soubor s dokumentací v podsložce `Doc/html` složky, do níž instalovali *Python*. Její podobu po „spuštění“ souboru `index.html` si můžete prohlédnout na obrázku 1.3.

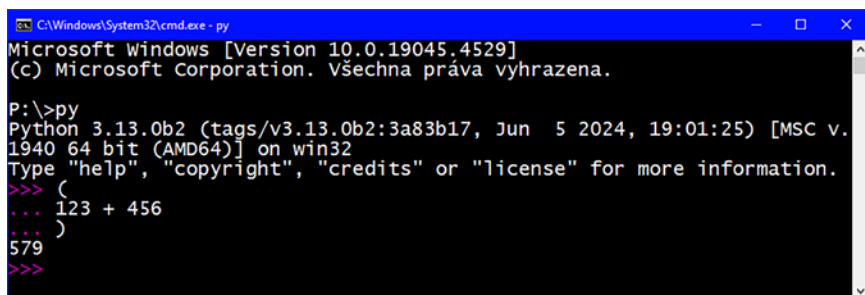


Obrázek 1.3: Okno s dokumentací platformy

Dokumentace je sice pouze anglicky, ale v současné době již existují poměrně kvalitní automatické překladače, které těm méně zkušeným požadovaný text přeloží.

1.2 Řádkový interpret

Základní komunikace s interpretem *Pythonu* probíhá přes příkazový řádek. Jak už jsem řekl, řádkový interpret *Pythonu* spustíme ve *Windows* zadáním příkazu **py** (viz obrázek 1.4) nebo **python**, na *macOS* a v *Linuxu* zadáním příkazu **python3**.



```
C:\Windows\System32\cmd.exe - py
Microsoft Windows [Version 10.0.19045.4529]
(c) Microsoft Corporation. Všechna práva vyhrazena.

P:\>py
Python 3.13.0b2 (tags/v3.13.0b2:3a83b17, Jun  5 2024, 19:01:25) [MSC v.
1940 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> (
... 123 + 456
... )
579
>>>
```

Obrázek 1.4: Spuštění interpretu z příkazového řádku Windows

Po spuštění se zobrazí úvodní řádek oznamující spuštění interpretu, doplněné informací o jeho verzi (řádek je dlouhý, tak se na obrázku zlomil na dva). Za ním je řádek informující o příkazech, po jejichž zadání získáte některé podrobnější informace.

Za těmito úvodními řádky následuje **výzva k zadání příkazu** (anglicky prompt). Tvoří ji čtveřice znaků „>>>“ (tři většítka následovaná mezerou). Když za výzvu něco napíšete a stisknete ENTER, interpret zadaný řádek zpracuje.

Pokud se interpret domnívá, že zadaný příkaz není kompletní a máte ještě něco zadat, objeví se na dalším řádku **pokračovací výzva** „...“, kterou tvoří tři tečky následované mezerou. Po ní interpret očekává zadání další části příkazu ukončené stiskem ENTER.

Na obrázku 1.4 je po výzvě zadána otevírací závorka, takže interpret si zapamatoval dosavadní zadání a vypsál pokračovací výzvu. Za ní jsem zadal výraz

```
123 + 456
```

Protože jsem nezadal zavírací závorku, tak si jej interpret po stisku ENTER jenom zapamatoval a na další řádek opět vypsál pokračovací výzvu.

Nyní jsem zadal uzavírací závorku a stiskl ENTER. Interpret zjistil, že jsem konečně zadal smysluplný příkaz, takže jej provedl, a protože se jednalo o výraz, který měl spočítat, vypsál na další řádek výsledek.

Pak opět vypsál úvodní výzvu k zadání příkazu a očekával zadání dalšího příkazu.



V dalším textu již nebudu zobrazovat komunikaci s počítačem prostřednictvím obrázků, ale prostřednictvím přehlednějších textových výpisů, jejichž kopie naleznete v souborech s doprovodnými programy.

1.2.1 Odsazování a pokračovací řádky

V *Pythonu* je (na rozdíl od většiny ostatních programovacích jazyků) důležité, jak moc daný řádek kódu odsadíte od levého kraje. Odsadíte-li jej méně nebo více, než je v danou chvíli potřeba, překladač se vzbouří a ohlásí syntaktickou chybu, tj. prohřešek proti pravidlům zápisu jazyka.

Příkaz zadávaný za výzvou `>>>` musí vždy začínat hned za výzvou bez jakýchkoliv úvodních mezer či tabulátorů. To je předvedeno v následujícím výpisu na řádce 1, kde jsem zadal součet čtyř čísel, na němž jsem ukázal, že na počtu případných mezer mezi jednotlivými součástmi příkazu nezáleží.

Pokud bychom však na začátek řádku vložili byť jedinou mezeru, jak je naznačeno na řádce 3, interpret ohlásí syntaktickou chybu *nečekané odsazení* (`unexpected indent`).

```
1 >>> 12+ 24 + 56 + 78
2 170
3 >>> 0
4 File "<unknown>", line 1
5 0
6 IndentationError: unexpected indent
7 >>>
```

Nevejde-li se nám zadávaný příkaz na řádek, musíme interpretu nějakým způsobem oznámit, že budeme pokračovat na následujícím řádku. Zadáváme-li nějaký delší výraz, můžeme jej například uzavřít do závorek, jak jsem to udělal na obrázku 1.4. Dokud v *Pythonu* neuzavřeme otevřenou závorku, bude *Python* považovat následující řádky za pokračovací až do chvíle, kdy tuto závorku neuzavřeme.

V pokračovacích řádcích *Python* odsazení od levého kraje nehlídá, takže v nich můžete text zarovnávat tak, aby byl pro nás co nejpřehlednější, jako je tomu v následujícím výpisu.

```

1 >>> ( 123
2 ...   + 456
3 ...   ) - 789
4 -210
5 >>>

```

Druhou možností, jak interpretu vysvětlit, že budete pokračovat na dalším řádku, je ukončit řádek znakem `\` (zpětné lomítko, anglicky *backslash*). Kdybychom při zápisu předchozího příkladu použili zpětná lomítka, vypadal by výpis následovně:

```

1 >>> 123 \
2 ...   + 456\
3 ...   - 789
4 -210
5 >>>

```

Všimněte si, že v obou případech se výsledek vypisuje od začátku řádku, kdežto všechny řádky příkazů jsou uvozené počáteční nebo pokračovací výzvou.



Jak jste si všimli, výzvy ve výpisech programů podbarvují jinou barvou než zbytek programu a zadávaný kód vysazují tučně, kdežto odpovědi programu obyčejně. Je to proto, abyste co nejsnadněji odlišili zadávaný kód od výzev interpretu.

Shrnutí a soubory pro opakování

V této kapitole jsme spustili pouze jednoduchý demonstrační skript uložený v souboru `m01a_s_cript.py`. Obsah jediného výpisu je v souborech s názvem `m01__Programy`. (Připomínám, že čistý zdrojový text je v souboru `m01__Programy.py` a text doplněný o čísla řádků z výpisů je v souboru `m01__Programy.pydoc`.)

Vážení čtenáři, právě jste dočetli ukázkou z knihy ***Python snadno a rychle***.
Pokud se Vám ukázka líbila, na našem webu si můžete zakoupit celou knihu.