Multimediální průvodce základů jazyka Python

Jan Horák

Bakalářská práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jan Horák**
Osobní číslo:	**A20364**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Multimediální průvodce základů jazyka Python**
Téma práce anglicky:	**A Multimedia Guide to the Basics of Python**

Zásady pro vypracování

1. Proveďte literární průzkum z oblasti jazyka Python a software použitého při tvorbě multimediálního průvodce.
2. Vypracujte sadu 14 výukových balíků pro výuku základů programování v jazyce v Python.
3. Vytvořte multimediálního průvodce na základě výukových balíků vytvořených dle zásady č. 2.
4. V praktické části práce popište vytvořené podklady v Python a multimediálního průvodce.
5. Vytvořte vzorová zadání a vypracování ukázkových cvičení.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. PECINOVSKÝ, Rudolf. Začínáme programovat v jazyku Python. 2. přepracované a rozšířené vydání. Praha: Grada Publishing, 2022. ISBN 978-80-271-3609-4.
2. PECINOVSKÝ, Rudolf. Python – knihovny pro práci s daty pro verzi 3.11. 1. vydání. Praha: Grada Publishing, 2022. ISBN 978-80-271-0659-2.
3. SUMMERFIELD, Mark. Python 3: výukový kurz. Brno: Computer Press, 2010. ISBN 978-80-251-2737-7.
4. PILGRIM, Mark. Ponořme se do Python(u) 3: Dive into Python 3. Praha: CZ.NIC, 2011. CZ.NIC. ISBN 978-80-904248-2--1.
5. DEDOV, Florian. The Python Bible 3 in 1: Volumes One to Three (Beginner, Intermediate, Data Science). Independently Published, 2019. ISBN 978-1089201847.

Vedoucí bakalářské práce: **Ing. Karel Perůtka, Ph.D.**
Ústav řízení procesů

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 22.5.2023

Jan Horák, v.r.

…….

podpis studenta

ABSTRAKT

Cílem této bakalářské práce je vytvoření čtrnácti výukových sad (balíků), určených primárně pro výuku předmětu zabývajícího se základy jazyka Python na vysokých školách. Jednotlivé výukové sady pokrývají konkrétní témata zadané vyučujícím dle jeho požadavků. Vypracované sady mají ustálenou strukturu, ve které je vždy brán ohled na srozumitelnost a přehlednost. Vypracované témata pokrývají oblasti jako syntaxe, datové typy, cykly a další základy nutné pro výuku tohoto jazyka. Balíky obsahují jak slovní popis, tak i ukázky kódu vztahující se k dané problematice. Součástí výstupu práce je i multimediální průvodce vytvořený sloužící k lepšímu pochopení probírané látky.

Klíčová slova: Python, výuka, multimediální

ABSTRACT

The goal of this bachelor's thesis is the creation of fourteen teaching sets (packages), intended primarily for teaching a subject dealing with the basics of the Python language at universities. Individual teaching sets cover specific topics specified by the teacher according to his requirements. The developed sets have a fixed structure, in which comprehensibility and clarity are taken in account. The developed topics cover areas such as syntax, data types, cycles and other basics necessary for learning this language. The packages contain both a verbal description and code samples related to the given topic. Part of output of the work is also a multimedia guide created for a better understanding of the discussed topic.

Keywords:Python, teaching, multimedia

Rád bych poděkoval Ing. Karlu Perůtkovi, Ph.D. za odbornou konzultaci a poskytnuté rady při tvorbě této bakalářské práce.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Python je v současné době jedním z nejpopulárnějších programovacích jazyků na světě a využívá se pro mnoho různých účelů, od vědeckých výpočtů až po tvorbu webových aplikací. Jeho popularity stále roste, a proto je důležité, aby byly zajištěny dostatečné zdroje pro výuku jeho základů. Cílem této práce je vytvořit multimediální průvodce, který bude sloužit jako zdroj pro začátečníky v oblasti jazyka Python. Tento průvodce bude obsahovat video tutoriály a PowerPoint prezentace, které budou pomáhat novým uživatelům Pythonu rychle porozumět jeho základům. V současné době existuje mnoho zdrojů, které se věnují výuce Pythonu, ale většina z nich se zaměřuje na výuku jeho pokročilejších funkcí. Základy jsou často přehlíženy a není dostatečně zdůrazněno, jak jsou důležité pro pochopení pokročilejších konceptů. Multimediální průvodce, který navrhuji, by mohl být snadno dostupný a interaktivní, což by mohlo pomoci při jeho výuce. Očekávaným přínosem této práce je tedy poskytnutí nového a efektivního zdroje pro výuku základů Pythonu a vyplnění mezery v existujících zdrojích výuky.

I. TEORETICKÁ ČÁST

1 ANALÝZA PROBLEMATIKY A MOŽNÝCH ŘEŠENÍ

Tato sekce zabývá popisem problematiky výuky programovacího jazyka Python na vysoké škole a shrnutím cílů této bakalářské práce.

1.1 Popis problematiky výuky na vysoké škole

Výuka jazyka Python na vysoké škole, která ho předtím nevyučovala, ale přesto ho využívala v jiných předmětech, může být problematická z několika důvodů. Za prvé, studenti na této škole mohou být neznalí jazyka Python a mohou mít omezené znalosti v oblasti programování obecně. To může způsobit zpomalení výuky, protože studenti budou muset najednou zvládnout nový jazyk a učit se nové koncepty programování. Za druhé, i když se jazyk Python využívá v jiných předmětech, jeho vyučování jako samostatného předmětu může být odlišné. Může být potřeba zahrnout více teoretického vzdělávání, například popis základní syntaxe a datových typů v jazyce Python. Navíc se mohou studenti, kteří se s jazykem setkali v jiných předmětech, naučit pouze omezený počet funkcí jazyka Python, což by mohlo být problematické.

Další výzvou může být nalezení kvalifikovaných učitelů, kteří mají dostatečné znalosti jazyka Python a zkušenosti s výukou na vysoké škole. Pokud takoví učitelé nejsou k dispozici, může být obtížné zajistit kvalitní výuku jazyka Python.

V poslední řádě je nutné zvážit, jaký druh materiálů a prostředků bude k dispozici pro výuku jazyka Python. Na vysoké škole by měly být k dispozici dostatečné zdroje, aby studenti mohli získat komplexní a kvalitní vzdělání jazyka Python.

1.2 Analýza literárních zdrojů

Analýza literárních zdrojů jazyka Python ukazuje, že existuje řada knih, online kurzů a dokumentace, které poskytují ucelené informace o tomto jazyce programování. Mezi nejvýznamnější zdroje patří oficiální dokumentace Pythonu, která obsahuje podrobné informace o syntaxi jazyka, vestavěných funkcích, standardní knihovně a dalších aspektech. Kromě toho existuje také řada knih, jako například „Learning Python“ od Marka Lutz, která je považována za klasiku mezi knihami věnovanými tomuto jazyku. Dalšími užitečnými zdroji jsou webové stránky jako Stack Overflow, GitHub a různá komunitní fóra, kde mohou uživatelé získat pomoc a podporu při programování v Pythonu. Kromě toho existují také online kurzy a školení, které umožňují získat hlubší znalosti o Pythonu. Nicméně,

mnoho z těchto zdrojů je psáno formou textových dokumentů, které mohou být pro začátečníky ob-tížně srozumitelné a neintuitivní. Tento problém se může projevit zejména u studentů, kteří se teprve učí programovat a nemají dostatek zkušeností s používáním programovacích jazyků.

1.3 Cíle a přínosy práce

Hlavním cílem je představit klíčové koncepty jazyka Python a pomoci studentům získat potřebné základy pro úspěšné programování. Mezi další cíle patří: Představit nejčastější programovací koncepty v jazyce Python jako proměnné, podmínky, cykly a funkce. Ukázat, jak Python může být použit pro řešení praktických úloh a problémů. Prezentovat různé nástroje a knihovny v jazyce Python a ukázat, jak je lze použít pro konkrétní úkoly. Přínosem této práce je, že umožní studentům naučit se jazyk Python vlastním tempem a v souladu se svým rozvrhem. Tato práce také pomáhá zvyšovat kvalitu výuky jazyka Python na vysokých školách, zejména na těch, které ho předtím nevyučovaly a používaly ho pouze v jiných předmětech.

1.4 Cílová skupina

Cílová skupina této bakalářské práce jsou studenti a vývojáři se zájmem o programování v jazyce Python. Práce je zaměřena na poskytnutí uceleného a strukturovaného výukového materiálu, který jim pomůže získat základní znalosti a dovednosti v Pythonu. Cílem je oslovit jak začátečníky, kteří se teprve seznamují s programováním, tak i pokročilé uživatele, kteří chtějí rozšířit své znalosti a dovednosti v Pythonu. Bakalářská práce má za úkol představit Python jako univerzální a mocný programovací jazyk a pomoci čtenářům porozumět jeho základním konceptům a principům.

2 SOFTWARE POUŽITÝ PŘI TVORBĚ MULTIMEDIÁLNÍHO PRŮVODCE

2.1 PowerPoint

Jedním ze softwarů použitých pro vytvoření multimediálního průvodce je PowerPoint. PowerPoint je prezentační software z produkce společnosti Microsoft. Tento software umožňuje uživatelům vytvářet prezentace, které mohou obsahovat textové, grafické a multimediální prvky. Pro tvorbu multimediálního průvodce základů jazyka Python lze využít právě tento nástroj. Uživatel může vytvářet snímky, na které může vkládat různé prvky jako jsou obrázky, grafy, videa nebo zvukové soubory a doplňovat je textem a odkazy na relevantní informace. PowerPoint je výhodný tím, že poskytuje uživatelům snadný způsob tvorby prezentací s atraktivními a interaktivními prvky a lze jej snadno ovládat. Výsledkem práce v Powerpointu je prezentace v podobě souboru, který lze snadno sdílet a použít jako multimediální průvodce. Samotná prezentace však nemusí být interaktivní, což může být nevýhodou, pokud si uživatel přeje aktivně pracovat s materiálem.

2.2 OBS

Open Broadcaster Software (OBS) je svobodný a otevřený software, který slouží ke záznamu a streamování videa. Je k dispozici pro operační systémy Windows, Mac OS X a Linux. Hlavním využitím OBS je nahrávání a streamování her a dalšího obsahu z počítačové obrazovky, ale může být také použit k nahrávání videí z webových kamer a dalších video vstupů. Mezi hlavní funkce OBS patří snadné nastavení záznamu a streamování, přizpůsobení vzhledu streamu pomocí scén a zdrojů, přidání efektů a filtrů, a možnost použití externích pluginů pro rozšíření funkcí. OBS také podporuje různé formáty videa, včetně standardních formátů jako MP4 a FLV, a umožňuje ukládat záznamy do lokálního úložiště nebo streamovat přímo na různé platformy, jako jsou YouTube, Twitch a další. Celkově je OBS užitečným nástrojem pro nahrávání a streamování videa a je oblíbený mezi streamery a tvůrci obsahu.

2.3 OpenShot

OpenShot je multiplatformní, open-source software pro úpravu videa. Je navržen pro snadné a intuitivní stříhání, přidávání efektů a tvorbu profesionálně vypadajících videí. OpenShot nabízí širokou škálu funkcí, včetně střihu videa, přidávání hudby a zvukových efektů,

tvorby přechodů, titulků a grafiky. Podporuje různé formáty videa a zvuku, a umožňuje export videa do různých populárních formátů. Díky jednoduchému uživatelskému rozhraní je OpenShot přístupný i pro začátečníky, ale zároveň nabízí dostatečné funkce a možnosti pro pokročilé uživatele. Je možné pracovat s vrstvami, upravovat klíčové snímky, aplikovat efekty jako zpomalení, zrychlení, zesvětlení nebo ztmavení videa a mnoho dalšího. OpenShot je dostupný pro operační systémy Windows, Mac OS X a Linux, což umožňuje jeho použití na různých platformách. Jeho otevřený zdrojový kód umožňuje komunitě přispívat k jeho vývoji a rozšiřovat jeho funkcionality. Celkově je OpenShot vhodným nástrojem pro úpravu videa, nejen pro profesionály, ale i pro amatérské tvůrce obsahu, kteří chtějí vytvářet atraktivní videa s profesionálním vzhledem.

3 PROGRAMOVACÍ JAZYK PYTHON

3.1 Co je to Python

Python je vysokoúrovňový, interpretovaný programovací jazyk, který se vyznačuje snadnou čitelností a intuitivitou. Byl vytvořen jako nástroj pro zpracování textu a automatizaci úloh. V současné době je Python jedním z nejpopulárnějších programovacích jazyků na světě. Python se vyznačuje jednoduchou syntaxí, což znamená, že kód je snadno čitelný a srozumitelný i pro začátečníky. Tento jazyk podporuje mnoho programovacích paradigmat, včetně procedurálního, objektově orientovaného a funkcionálního programování. Jeho velkou výhodou je také široká dostupnost knihoven a modulů, které usnadňují vývoj a šetří čas programátorům. Python je rovněž multiplatformní, tedy funkční na většině operačních systémů, což usnadňuje jeho použití pro různé účely. Python také disponuje velkou a aktivní komunitou, která vytváří knihovny, moduly a nástroje pro vývoj a podporu tohoto jazyka. Celkově lze tedy říci, že Python je univerzální a flexibilní programovací jazyk, který díky své jednoduchosti a množství dostupných nástrojů usnadňuje a urychluje vývoj aplikací v různých oblastech.[1]

3.2 Historie

Python je interpretovaný programovací jazyk, který byl vyvinut v roce 1989 Guidem van Rossumem v Centru pro matematiku a informatiku (CWI) v Nizozemsku. Jazyk byl původně navržen jako nástroj pro automatizaci různých úloh a psaní skriptů. První veřejně dostupná verze Pythonu, verze 0.9.0, byla vydána v roce 1991. Od té doby bylo vydáno mnoho dalších verzí jazyka, včetně současné stabilní verze Pythonu 3.10. Historie jazyka Python je plná inovací a zlepšení, jako například přechod na podporu Unicode v Pythonu 3 a výrazné zlepšení výkonu interpretu. Díky své jednoduché syntaxi, čitelnosti a rozsáhlým knihovnám se Python stal jedním z nejoblíbenějších programovacích jazyků na světě a nachází široké uplatnění v mnoha oblastech, včetně datové analýzy, strojového učení, webového vývoje a automatizace úloh.[2]

3.2.1 Python 1

První verze jazyka Python byla vydána v roce 1991. Python 1.0 byl původně navržen jako jednoduchý a snadno čitelný skriptovací jazyk s důrazem na modularitu a konsistenci syntaxe. Jazyk byl vytvořen jako nástupce jazyka ABC. Python 1.0 byl poměrně jednoduchý a

obsahoval pouze několik základních datových typů, výrazy a funkce, ale již tehdy byl kladen důraz na čitelnost kódu a modularitu. V první verzi Pythonu byly implementovány některé klíčové funkce, jako například definice funkcí, podmínky a cykly. V této verzi jazyka Python byly také zavedeny třídy a objekty, které umožňovaly programátorům pracovat s vysokoúrovňovými abstrakcemi dat. Tato verze Pythonu byla také první, která byla volně šiřitelná pod licencí GPL (General Public License). Python 1.0 byl první verzí jazyka, který měl nějaký rozsah a uživatelskou základnu, a položil základy pro další vývoj jazyka.[2]

3.2.2 Python 2

Python 2 je druhá generace jazyka Python, která byla vydána v roce 2000. Tato verze Pythonu přinesla mnoho nových vlastností, včetně podpory Unicode, nových datových typů, rozšíření modulu pro práci se sítí a mnoho dalšího. Python 2 také zahrnoval novou implementaci interpretu jazyka, označenou jako CPython, která byla zcela napsána v jazyce C. Nicméně, v roce 2008 byla vydána nová verze Pythonu, Python 3, která přinesla mnoho vylepšení, včetně nových funkcí a vylepšeného chování existujících funkcí. Kvůli těmto změnám však Python 3 není zpětně kompatibilní s Pythonem 2, což způsobilo určité problémy pro programátory a společnosti, které byly závislé na starších verzích Pythonu. V současné době Python 2 již není oficiálně podporován a vývoj se soustředí na novější verze jazyka.[2]

3.2.3 Python 3

Python 3 je nejnovější verze programovacího jazyka Python, která byla vydána v roce 2008. Tato verze zahrnuje několik významných změn a novinek oproti předchozí verzi Pythonu 2. Jednou z nejvýznamnějších změn je změna chování výchozího kódování na UTF-8 a rozšíření podpory pro Unicode. Dalšími významnými změnami jsou například nové datové typy, jako jsou například bytové řetězce, nové funkce pro manipulaci s řetězci a nový způsob formátování řetězců. Python 3 také přináší významné změny v syntaxi jazyka. Mezi nejvýznamnější patří změny ve způsobu používání print funkce, změny v chování při dělení celočíselných hodnot a nová syntaxe pro definici funkce. Tyto změny byly provedeny s cílem zlepšit čitelnost kódu a zjednodušit jeho psaní. Python 3 také přináší mnoho nových knihoven a modulů, které usnadňují vývoj aplikací v Pythonu. Mezi nejvýznamnější patří například knihovny pro práci s asynchronním programováním, pro práci s daty a časem, pro práci s regulárními výrazy, pro práci s grafikou a mnoho dalších. Celko-

vě lze říci, že Python 3 je významným krokem vpřed pro programovací jazyk Python a přináší mnoho vylepšení, nových funkcí a knihoven. Přestože je tato verze zpětně nekompatibilní s předchozí verzí Pythonu 2, stále se stává stále populárnější volbou pro vývoj moderních aplikací.[2]

3.3 Využití

Python se díky své jednoduchosti, čitelnosti a rozmanitosti používá v mnoha oblastech. Patří mezi ně například: Webové aplikace a serverové aplikace, Data science a strojové učení, Automatizace a scripting, Desktopové aplikace, Hry a vizualizace. Jelikož je Python velmi výkonný a flexibilní jazyk, jeho využití je velmi široké a může být adaptován pro různé účely a oblasti.[3]

3.4 Vývojová prostředí

Přestože Python nabízí mnoho funkcí a knihoven, které umožňují vytvoření složitých aplikací, vývojáři stále potřebují nástroje, které jim umožní psát, testovat a ladit kód efektivněji. Vývojové prostředí pro Python jsou softwarové nástroje, které pomáhají vývojářům při psaní, testování a ladění Python kódu. Tyto nástroje zahrnují textové editory, integrovaná vývojová prostředí (IDE) a webové rozhraní. Python může být spouštěn z příkazové řádky, ale tento způsob psaní kódu je pomalý a neefektivní. Vývojové prostředí umožňují vývojářům psát kód rychleji a efektivněji díky funkci automatického dokončování kódu, syntaxe highlightingu, možnostem refraktorování a mnoha dalším.[4]

3.4.1 IDLE

IDLE (Integrated Development and Learning Environment) je vývojové prostředí (IDE) pro Python, které je součástí jeho základní instalace. Jedná se o jednoduché a intuitivní prostředí, které je vhodné pro začátečníky v programování. IDLE obsahuje mnoho funkcí, jako například interaktivní shell pro okamžité spouštění kódu, editor kódu s podporou zvýrazňování syntaxe, automatické doplňování kódu a možnosti spouštění skriptů. IDLE je výbornou volbou pro jednoduché projekty, jako například skripty pro automatizaci úloh, programy pro zpracování dat nebo jednoduché webové aplikace. Pro složitější projekty je však lepší použít pokročilejší vývojové prostředí, jako například PyCharm nebo VS Code.[5]

3.4.2 VS Code

Pro vývoj v Pythonu existuje mnoho různých vývojových prostředí, které mohou být použity pro psaní a spouštění kódu. Jedním z nejpopulárnějších vývojových prostředí pro Python je VS Code, který je nabízen společností Microsoft. VS Code poskytuje mnoho užitečných funkcí pro vývoj v Pythonu, jako je syntax highlighting, autocompletion, debugging a integraci s verzovacími systémy. Dalším důvodem, proč si zvolit VS Code pro vývoj v Pythonu, je rozšiřitelnost. VS Code je vybaven mnoha rozšířeními, které mohou být instalovány a použity pro zlepšení produktivity a poskytnutí dalších užitečných funkcí. Existuje také mnoho rozšíření specifických pro Python, jako například rozšíření pro vývoj Flask a Django aplikací. Kromě toho je VS Code multiplatformní a lze ho použít na různých operačních systémech, jako jsou Windows, Linux a macOS. To znamená, že vývojáři mohou používat stejné vývojové prostředí na různých počítačích a operačních systémech.[6]

3.4.3 PyCharm

PyCharm je integrované vývojové prostředí (IDE) pro jazyk Python, vyvinuté společností JetBrains. Toto IDE obsahuje nástroje pro zjednodušení procesu vývoje v Pythonu, jako je například integrace s verzovacími nástroji, laděním kódu a podporou testování. PyCharm nabízí mnoho funkcí pro usnadnění práce s Pythonem, jako je například inteligentní kódování, které usnadňuje psaní kódu a zabraňuje chybám, funkce pro refactorování kódu a podpora pro tvorbu grafických uživatelských rozhraní (GUI). PyCharm má také integrovaný správce balíčků, který umožňuje snadno instalovat a aktualizovat balíčky Pythonu. Kromě toho PyCharm nabízí rozšíření pro různé další technologie, jako je například Django, Flask, Pyramid, a další.[7]

3.4.4 Spyder

Vývojové prostředí Spyder je open-source multiplatformní IDE (Integrated Development Environment) navržené pro vědecké výpočty a analýzy dat v jazyce Python. Spyder nabízí přehledné rozhraní pro programování a umožňuje uživatelům vytvářet, testovat a ladit kód rychle a efektivně. Mezi hlavní vlastnosti patří interaktivní konzole, která umožňuje spouštět kód postupně, plnou integraci s IPython konzolí, snadný přístup k dokumentaci a možnost vytváření grafů a vizualizací dat. Spyder je také vybaven podporou pro správu projektů, což umožňuje uživatelům snadno organizovat svůj kód a spravovat různé verze svých

projektů. Spyder se často používá pro vědecké výpočty a analýzy dat, ale je také vhodný pro běžné úkoly, jako je tvorba webových aplikací a automatizace úloh. Díky svému snadno použitelnému rozhraní a množství funkcí je Spyder populárním vývojovým prostředím pro Python. Spyder nabízí více funkcí zaměřených na analýzy dat, ale na druhé straně má méně pokročilých nástrojů pro vývoj webových aplikací v porovnání s předešle zmíněnými vývojovými prostředí.[8]

3.4.5 Jupyter

Jupyter je interaktivní vývojové prostředí (IDE), které umožňuje vytváření a sdílení dokumentů nazývaných „notebooky“. Jupyter kombinuje kód, textové poznámky a výstupy do jednoho dokumentu, což umožňuje snadné experimentování, vizualizaci dat a sdílení výsledků. Jupyter podporuje různé programovací jazyky, ale je nejčastěji používán s jazykem Python. Jupyter poskytuje prostředí pro interaktivní programování, analýzu dat, vědecký výzkum a vývoj prototypů. Umožňuje vám psát kód ve svých buňkách a okamžitě vykonávat tento kód, což usnadňuje rychlé testování a iteraci. S Jupyterem můžete také vkládat textové poznámky, matematické vzorce, obrázky a interaktivní prvky do svých notebooků, což vám umožňuje dokumentovat svůj kód a sdílet své myšlenky a výsledky. Jupyter je široce používán v oblastech jako datová analýza, strojové učení, výzkum a vzdělávání. Jeho interaktivní povaha a schopnost kombinovat kód, text a vizualizace poskytují vývojářům a výzkumníkům mocný nástroj pro efektivní práci s daty a rychlé prototypování nových myšlenek.[9]

3.5 Výběr vývojového prostředí

Výběr správného vývojového prostředí může být pro začínající programátory a studenty obtížný úkol. Existuje mnoho vývojových prostředí dostupných pro programování v jazyce Python, a každé z nich má své výhody a nevýhody. Různá prostředí se liší funkcionalitou, uživatelským rozhraním, podporovanými funkcemi a také tím, jak se hodí k určitým projektům a osobním preferencím. Při výběru vývojového prostředí je důležité zvážit několik faktorů. Patří sem například podpora jazyka Python, dostupnost potřebných funkcí a nástrojů, snadnost použití, rozšiřitelnost prostředí, komunitní podpora a samozřejmě také osobní preference. V případě výběru prostředí VS Code byla brána v úvahu jeho dostupnost na fakultě aplikované informatiky UTB ve Zlíně, což je pro studenty výhodné, protože se mohou seznámit s tímto prostředím již během studia. Navíc, VS Code je multiplatform-

ní a lze ho nainstalovat na vlastní zařízení, což umožňuje studentům pokračovat ve vývoji i mimo fakultní prostředí. Tato dostupnost a konzistence umožňují studentům pohodlně pracovat s Pythonem nezávisle na zařízeních a prostředí. Výběr vývojového prostředí je často subjektivní záležitostí a závisí na individuálních preferencích a potřebách uživatele. Důležité je najít prostředí, které splňuje požadavky, umožňuje efektivní vývoj a zároveň je příjemné a pohodlné pro práci.

3.6 Porovnání jazyka Python s ostatními

Python je výkonný programovací jazyk, který má své výhody a nevýhody ve srovnání s ostatními programovacími jazyky.

Výhody[10]:

- Jednoduchá syntaxe: Python má čistou a jednoduchou syntaxi, která usnadňuje čtení a psaní kódu. To znamená, že je snadný pro začátečníky a umožňuje rychlé prototypování.
- Velká komunita a podpora: Python má obrovskou komunitu vývojářů, kteří sdílejí své znalosti a zkušenosti. Existuje mnoho online zdrojů, knihoven a frameworků, které usnadňují vývoj a řešení různých problémů.
- Multiplatformní podpora: Python je křížově-platformní jazyk, což znamená, že může být spuštěn na různých operačních systémech, jako jsou Windows, macOS, Linux a další.
- Široká škála aplikací: Python je univerzální jazyk, který lze použít pro různé účely, jako je webový vývoj, datová analýza, strojové učení, automatizace, vědecký výzkum a další. Má bohatou sadu knihoven a nástrojů, které podporují tato různorodá použití.

Nevýhody[10]:

- Pomalejší než některé jiné jazyky: Python je interpretovaný jazyk, což znamená, že je obecně pomalejší než některé kompilované jazyky, jako je C nebo Java. To může být problematické pro aplikace, které vyžadují vysokou výpočetní rychlost.
- Omezená podpora pro mobilní vývoj: Python není primárním jazykem pro vývoj mobilních aplikací a má omezenou podporu v oblasti mobilních platforem. Pro vývoj mobilních aplikací se častěji používají jiné jazyky, jako je Java nebo Swift.

- Globální interpreter a řízení verzí: Python má své vlastní globální prostředí pro interpretaci kódu, což může vést k problémům s řízením verzí a kompatibilitou mezi různými verzemi Pythonu.
- Omezená podpora pro velmi nízkoúrovňové programování: Python není ideální volbou pro vývoj velmi nízkoúrovňových systémových aplikací, jako jsou operační systémy nebo vestavěné systémy.

4 ZÁKLADY PROGRAMOVÁNÍ V JAZYCE PYTHON

4.1 Datové typy

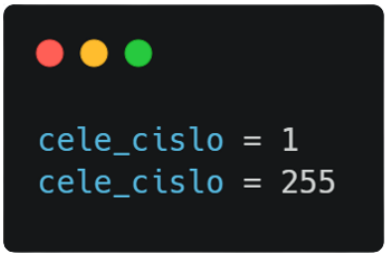
V Pythonu existuje několik základních datových typů, které se používají k reprezentaci různých druhů hodnot. Tyto datové typy zahrnují číselné typy, řetězce, logické hodnoty, seznamy, n-tice, množiny a slovníky. Každý datový typ má své vlastnosti a metody, které umožňují manipulaci s daty.[11]

4.1.1 Číselné typy

V jazyce Python existují různé číselné typy, které se používají pro reprezentaci různých druhů čísel. Tyto typy zahrnují celá čísla (*int*), desetinná čísla (*float*) a komplexní čísla (*complex*).[11]

4.1.1.1 Celá čísla

Celá čísla (*int*) jsou jedním z datových typů v jazyce Python, které slouží k reprezentaci celých čísel bez desetinné části.



Obrázek 1: Přiřazení celého čísla proměnné

V Pythonu jsou celá čísla základním a rozšířeným datovým typem, který je široce používán při programování. Python nepoužívá pevně definovaný rozsah pro celá čísla, což znamená, že můžeme použít velmi velká i velmi malá čísla v závislosti na dostupné paměti. V případě, že číslo přesahuje maximální hodnotu pro daný datový typ, Python automaticky převede číslo na datový typ *long*, který umožňuje reprezentovat i velmi velká celá čísla.[11]

4.1.1.2 Desetinná čísla

Desetinná čísla v jazyce Python jsou reprezentována pomocí datového typu *float*. Jsou určena pro reprezentaci čísel s desetinnou částí a umožňují práci s čísly s pohyblivou řádovou čárkou.

Obrázek 2: Přiřazení desetinného čísla proměnné

Desetinná čísla mohou mít jak kladnou, tak i zápornou hodnotu. Python používá standard IEEE 754 pro reprezentaci desetinných čísel s pohyblivou řádovou čárkou. Tento standard definuje přesnost a rozsah desetinných čísel v Pythonu. Nicméně, kvůli omezením tohoto standardu mohou být desetinná čísla v Pythonu občas ovlivněna zaokrouhlovacími chybami. Při provádění matematických operací s desetinnými čísly v Pythonu je důležité mít na paměti, že některé operace mohou vést k nepřesným výsledkům kvůli omezené přesnosti desetinných čísel. Je proto vhodné být opatrný při porovnávání desetinných čísel pomocí rovnosti (==) a zvažovat použití funkce *math.isclose()* pro porovnávání hodnot určitou tolerancí. [11, 12]

4.1.1.3 Komplexní čísla

Komplexní čísla v jazyce Python jsou reprezentována pomocí datového typu *complex*. Jsou používána pro práci s čísly obsahujícími jak reálnou, tak imaginární část. Komplexní čísla jsou ve tvaru $a + bj$, kde a je reálná část a b je imaginární část čísla.

Obrázek 3: Přiřazení komplexního čísla proměnné

Při manipulaci s komplexními čísly je možné využít matematické moduly, jako je například *cmath*, který poskytuje funkce pro komplexní aritmetiku a matematické operace s komplexními čísly. Python umožňuje využívat komplexní čísla při různých výpočtech, jako je například analýza signálů, výpočty elektromagnetických polí nebo v oblasti matematického modelování.

Python poskytuje bohatou podporu pro práci s číselnými datovými typy. Kromě základních operací jako sčítání, odčítání, násobení a dělení je také možné provádět pokročilé matematické operace a funkce. Díky dynamické typovosti Pythonu je také možné převádět číselné typy na sebe navzájem, pokud je to potřeba.[11]

4.1.2 Řetězce

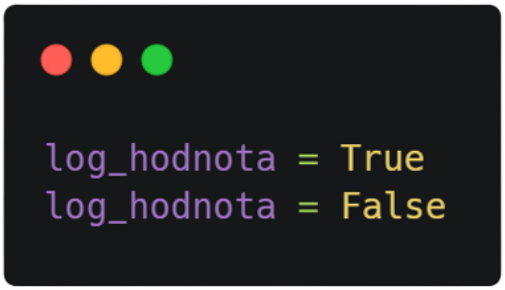
Řetězce (*strings*) jsou v jazyce Python reprezentovány jako sekvence znaků a slouží k manipulaci s textovými daty. Řetězce jsou v Pythonu immutable, což znamená, že po vytvoření nelze jejich obsah změnit. V Pythonu lze řetězce vytvořit pomocí jednoduchých (′), dvojitých (″) nebo trojitých (‴ nebo ″″″) uvozovek.

Obrázek 4: Přiřazení řetězce proměnné

Uvozovky lze použít volně, ale musí se dodržet stejný typ uvozovek na začátku a na konci řetězce. Při práci s řetězci lze využívat různé operace a metody. Mezi základní operace patří konkatenace (spojování řetězců) pomocí operátoru +, opakování řetězců pomocí operátoru (*), přístup k jednotlivým znakům pomocí indexování a řezání (slicing) řetězců. Python také poskytuje mnoho vestavěných metod pro manipulaci s řetězci. Některé z těchto metod zahrnují změnu velikosti písmen, vyhledávání podřetězců, nahrazování částí řetězce, odstraňování mezer, rozdělování řetězců na podřetězce a mnoho dalšího. Python také podporuje formátování řetězců pomocí různých metod, jako je například metoda *format()* nebo tzv. „*f-strings*“, které umožňují vkládat proměnné přímo do řetězců. Řetězce v Pythonu jsou velmi flexibilní a nacházejí široké uplatnění při práci s textovými daty, manipulaci s řetězci, parsování a analýze textových souborů, webovém skrapingu, zpracování řetězcových vstupů od uživatele a mnoho dalšího.[11]

4.1.3 Logické hodnoty

Logické hodnoty v Pythonu jsou reprezentovány jako *True* (pravda) nebo *False* (nepravda).



Obrázek 5: Přiřazení logické hodnoty proměnné

Tyto hodnoty se často používají při vyhodnocování podmínek a řízení toku programu. V Pythonu existuje také speciální datový typ nazývaný *bool*, který může mít pouze dvě hodnoty: *True* nebo *False*. Logické hodnoty jsou základním stavebním kamenem podmínek, cyklů a logických operací. Při vyhodnocování logických podmínek lze v Pythonu použít relační operátory, jako je rovnost (==), nerovnost (*!*=), menší než (<), větší než (>), menší nebo rovno (<=), větší nebo rovno (>=). Výsledkem takových porovnání je logická hodnota *True* nebo *False*. Python také poskytuje logické operátory, jako je logický součet (*or*), logický součin (*and*) a logický negace (*not*), které umožňují kombinovat logické hodnoty do složitějších podmínek. Logické hodnoty jsou velmi důležité při řízení toku programu a rozhodování o tom, která část kódu se má vykonat nebo vynechat na základě splnění určitých podmínek. [11]

4.1.4 Seznamy (Listy)

Seznamy jsou v Pythonu jedním z nejčastěji používaných datových typů. Jsou to závorky, do kterých můžete vložit libovolný počet hodnot oddělených čárkou.

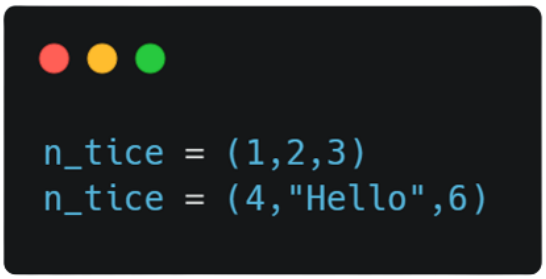
Obrázek 6: Přiřazení seznamu proměnné

Seznamy mohou obsahovat různé typy prvků, jako jsou čísla, řetězce, logické hodnoty nebo dokonce jiné seznamy. Seznamy v Pythonu jsou indexovány, což znamená, že jednotlivým prvkům můžeme přistupovat pomocí jejich pozice v seznamu. Indexování v Pythonu

začíná od 0, takže první prvek seznamu má index 0, druhý prvek index 1 a tak dále. Kromě indexování mohou seznamy v Pythonu také měnit svou délku a obsah. Můžete přidávat nové prvky na konec seznamu pomocí metody *append()*, vkládat prvky na konkrétní pozici pomocí metody *insert()*, odstraňovat prvky pomocí metody *remove()* nebo *del*, a mnoho dalšího. Pro práci se seznamy v Pythonu existuje mnoho vestavěných funkcí a metod, které usnadňují jejich manipulaci. Můžeme například spojovat seznamy pomocí operátoru +, opakovat prvky seznamu pomocí operátoru (*), zjistit délku seznamu pomocí funkce *len()*, nebo provádět různé operace s prvky seznamu pomocí metod, jako je *sort()*, *reverse()*, *index()* atd. Seznamy jsou velmi flexibilním a užitečným datovým typem v Pythonu, který se často používá pro ukládání a manipulaci s kolekcemi dat.[13]

4.1.5 N-tice (Tuple)

N-tice v Pythonu jsou datovým typem, který se podobá seznamům, ale s jedním významným rozdílem – jsou neměnné. To znamená, že po vytvoření n-tice nelze její obsah měnit. N-tice se často používají pro ukládání kolekcí hodnot, které mají pevně daný pořadí a nemění se. N-tice se v Pythonu vytvářejí pomocí kulatých závorek a hodnoty jsou odděleny čárkami.

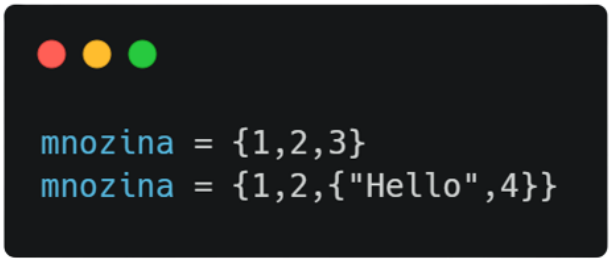


Obrázek 7: Přiřazení n-tice proměnné

Na rozdíl od seznamů mohou n-tice obsahovat prvky různých datových typů, jako jsou čísla, řetězce, logické hodnoty nebo dokonce jiné n-tice. K přístupu k prvkům n-tice slouží indexování, které funguje stejným způsobem jako u seznamů. Indexy začínají od 0 a lze použít k přístupu k jednotlivým prvkům nebo pro provedení různých operací, jako je například vyjmutí podřetězce. Protože jsou n-tice neměnné, nelze je přímo měnit nebo rozšiřovat. Pokud je potřeba provést změny na kolekci hodnot, kterou lze upravovat, je lepší použít seznamy. N-tice mají své využití zejména tam, kde je potřeba ukládat data, která mají neměnný charakter, jako jsou souřadnice bodů, informace o kalendáři, časová razítka atd.[14]

4.1.6 Množiny (Set)

Množiny v Pythonu jsou datovým typem, který slouží ke shromažďování jedinečných prvků. Množiny se vytvářejí pomocí složených závorek {} a prvky jsou odděleny čárkami.



Obrázek 8: Přiřazení množiny proměnné

Jednou z klíčových vlastností množin je, že každý prvek v množině je jedinečný, což znamená, že se v množině nemůže vyskytovat vícekrát. Existují různé operace, které lze provádět s množinami. Mezi základní operace patří přidávání prvků do množiny, odstraňování prvků z množiny, sjednocení množin, průnik množin, rozdíl množin a další. Python poskytuje vestavěné metody pro tyto operace, které lze použít na množinách. Množiny mají různá využití v programování. Mohou být použity pro odstranění duplicitních prvků z kolekce, pro kontrolu přítomnosti prvků nebo pro vykonávání matematických operací s množinami. Při práci s množinami je důležité mít na paměti, že množiny jsou neuspořádané, což znamená, že nemají pevné pořadí prvků. To také znamená, že nelze na množiny aplikovat indexování nebo vybrat prvek podle indexu.[14]

4.1.7 Slovníky (Dictionary)

Slovníky jsou v Pythonu datovým typem, který slouží ke shromažďování a organizaci dat ve formě klíč-hodnota. Každý prvek v slovníku se skládá z klíče a odpovídající hodnoty. Klíč je unikátní identifikátor, který slouží k přístupu k hodnotě. Hodnota může být libovolný objekt v Pythonu. Slovníky se vytvářejí pomocí složených závorek {} nebo pomocí funkce *dict()*.

Obrázek 9: Přiřazení slovníku proměnné

Každý prvek v slovníku je zapsán ve tvaru „klíč: hodnota“. Můžete přistupovat k hodnotám v slovníku pomocí klíče, což je efektivní způsob vyhledávání dat. Slovníky mají různá využití v programování. Mohou být použity k ukládání a správě dat, jako například seznam kontaktů, slovníky s překlady, konfigurační soubory a další. Díky rychlému vyhledávání pomocí klíčů jsou slovníky vhodné pro manipulaci s velkým množstvím dat. Python poskytuje mnoho vestavěných metod pro práci se slovníky. Můžete přidávat nové prvky, mazat prvky, upravovat hodnoty, zjišťovat délku slovníku, kontrolovat přítomnost klíčů a další operace. Při práci se slovníky je důležité mít na paměti, že slovníky jsou neuspořádané, což znamená, že nemají pevné pořadí prvků. To znamená, že při procházení slovníku pořadí prvků nemusí být zachováno.[14]

4.2 Proměnné

Proměnné v Pythonu jsou pojmenované úložiště, které slouží k uchovávání a manipulaci s daty. Každá proměnná má svůj název, kterým ji identifikujeme, a hodnotu, kterou obsahuje. Proměnné v Pythonu jsou dynamicky typované, což znamená, že nemusíme deklarovat typ proměnné před použitím a mohou měnit svůj typ v průběhu programu. Při vytváření proměnné v Pythonu se používá zápis „název = hodnota“. Název proměnné by měl být výstižný a snadno identifikovatelný, aby byl kód čitelný a srozumitelný. Hodnota proměnné může být libovolný objekt v Pythonu, jako například číslo, řetězec, seznam nebo dokonce jiná proměnná. Při přiřazování hodnoty proměnné se provádí operace přiřazení, která vytváří vazbu mezi názvem proměnné a její hodnotou. Pokud později změníme hodnotu proměnné, je automaticky aktualizována. Proměnné v Pythonu mají několik vlastností:

- Dynamické typování: Nemusíme explicitně deklarovat typ proměnné, protože Python automaticky rozpoznává typ na základě přiřazené hodnoty.
- Dynamická alokace paměti: Paměť pro proměnné se dynamicky alokuje a uvolňuje, což zjednodušuje správu paměti v porovnání s jinými jazyky.
- Dostupnost v různých rozsazích: Proměnné mohou být globální (přístupné ve všech částech programu) nebo lokální (omezené na určitou oblast, jako je funkce nebo třída).
- Flexibilita: Proměnné mohou měnit svůj typ v průběhu programu, což umožňuje různé operace a manipulace s daty.

Při práci s proměnnými je důležité mít na paměti některá pravidla, jako je platný formát názvu proměnné, správné použití přiřazovacího operátoru a zohlednění rozsahu platnosti proměnných.[15]

4.3 Operátory

Operátory v Pythonu jsou symboly, které umožňují provádět různé operace s daty. Python podporuje širokou škálu operátorů, které se používají pro aritmetické operace, porovnávání, logické operace, přiřazování hodnot a další manipulace s daty.

4.3.1 Aritmetické operátory

Aritmetické operátory v Pythonu slouží k provádění matematických operací s čísly. Python podporuje následující aritmetické operátory:

- Sčítání (+): Používá se k sčítání dvou čísel nebo k spojování řetězců. Příklad: 2 + 3 vrátí hodnotu 5, 'Hello ' + 'World' vrátí řetězec 'Hello World'.
- Odčítání (-): Používá se k odčítání jednoho čísla od druhého. Příklad: 5 - 3 vrátí hodnotu 2.
- Násobení (*): Slouží k násobení dvou čísel. Příklad: 2 * 3 vrátí hodnotu 6.
- Dělení (/): Provádí dělení jednoho čísla druhým. Výsledek je vždy v plovoucí desetinné čárce (*float*). Příklad: 7 / 2 vrátí hodnotu 3.5.
- Celočíselné dělení (//): Provádí dělení jednoho čísla druhým a vrátí celočíselný výsledek. Příklad: 7 // 2 vrátí hodnotu 3.
- Modulo (%): Vrací zbytek po celočíselném dělení dvou čísel. Příklad: 7 % 2 vrátí hodnotu 1.

- Exponenciální umocňování (**): Umocňuje první číslo na druhé. Příklad: 2 ** 3 vrátí hodnotu 8.

Je důležité si uvědomit, že při provádění aritmetických operací v Pythonu je třeba brát v potaz prioritu operátorů a používat závorky, pokud je to potřeba k dosažení požadovaného pořadí výpočtu.[16]

4.3.2 Porovnávací operátory

Porovnávací operátory v Pythonu slouží k porovnávání hodnot a vrací logickou hodnotu *True* nebo *False* podle výsledku porovnání. Python podporuje následující porovnávací operátory:

- Rovnost (==): Porovnává, zda jsou dva operandy rovny. Vrací *True*, pokud jsou hodnoty shodné, a *False*, pokud nejsou. Příklad: 5 == 5 vrátí hodnotu *True*.
- Nerovnost (!=): Porovnává, zda jsou dva operandy nerovné. Vrací *True*, pokud hodnoty nejsou shodné, a *False*, pokud jsou. Příklad: 5 != 3 vrátí hodnotu *True*.
- Větší než (>), Menší než (<): Porovnávají, zda je první operand větší než nebo menší než druhý operand. Vrací *True*, pokud je podmínka splněna, a *False*, pokud není. Příklad: 5 > 3 vrátí hodnotu *True*.
- Větší nebo rovno (>=), Menší nebo rovno (<=): Porovnávají, zda je první operand větší nebo roven než druhý operand, nebo menší nebo roven. Vrací *True*, pokud je podmínka splněna, a *False*, pokud není. Příklad: 5 >= 5 vrátí hodnotu *True*.

Porovnávací operátory lze používat nejen s číselnými hodnotami, ale také s řetězci, seznamy a dalšími datovými typy v Pythonu. Je důležité si uvědomit, že v případě řetězců je porovnání prováděno na základě lexikografického pořadí.[16]

4.3.3 Logické operátory

Logické operátory v Pythonu slouží k provádění logických operací mezi výrazy a vrací logickou hodnotu *True* nebo *False*. Python podporuje následující logické operátory:

- Logický součet (*or*): Vrací *True*, pokud alespoň jeden z operandů je *True*, jinak vrací *False*. Příklad: *True* or *False* vrátí hodnotu *True*.
- Logický součin (*and*): Vrací *True*, pouze pokud oba operandy jsou *True*, jinak vrací *False*. Příklad: *True* and *False* vrátí hodnotu *False*.

- Logická negace (*not*): Invertuje logickou hodnotu operandu. Vrací *True*, pokud je operand *False*, a *False*, pokud je operand *True*. Příklad: not *True* vrátí hodnotu *False*.

Logické operátory lze kombinovat a vytvářet složitější logické výrazy. Je také možné využívat závorky pro určení pořadí vyhodnocování.[16]

4.3.4 Přiřazovací operátory

Přiřazovací operátory v Pythonu slouží k přiřazení hodnoty do proměnné. Python používá následující přiřazovací operátory:

- Přiřazení (=): Přiřazuje hodnotu výrazu vpravo do proměnné vlevo. Příklad: x = 5 přiřadí hodnotu 5 proměnné x.
- Přiřazení s operací (+, -, *, /, atd.) (+=, -=, *=, /=, atd.): Přiřazuje hodnotu výrazu vpravo k hodnotě proměnné vlevo po provedení operace a výsledek opět přiřazuje proměnné. Příklad: x += 2 přičte 2 k hodnotě proměnné x a novou hodnotu přiřadí proměnné x.
- Přiřazení s rozbalováním (unpacking) (*=, /=, atd.): Přiřazuje hodnotu výrazu vpravo do proměnných vlevo, rozbaluje výraz na jednotlivé hodnoty a přiřazuje je do proměnných. Příklad: a, b = 1, 2 přiřadí hodnotu 1 proměnné a a hodnotu 2 proměnné b.

Přiřazovací operátory umožňují snadné a efektivní manipulace s proměnnými v Pythonu.[16]

4.4 Výrazy

Výrazy v Pythonu jsou kombinace proměnných, literálů, operátorů a funkcí, které vyjadřují nějakou hodnotu. Výrazy mohou být jednoduché nebo složené a mohou obsahovat aritmetické operace, porovnávání, logické operace, volání funkcí a další operace. Jednoduché výrazy jsou například aritmetické výrazy, jako je 2 + 3, který vyhodnotí sečet dvě čísla a vrátí výsledek 5. Dalším příkladem je porovnávací výraz x > y, který vyhodnotí, zda je proměnná x větší než proměnná y, a vrátí logickou hodnotu *True* nebo *False*. Složené výrazy jsou vytvořeny kombinací jednoduchých výrazů pomocí operátorů a funkcí. Například výraz 2 * (x + y) kombinuje operaci sčítání, násobení a závorky pro určení správného pořadí operací. Výrazy v Pythonu jsou vyhodnocovány na hodnoty pomocí operátorů a funk-

cí. Výsledek výrazu může být přiřazen do proměnné, použit v podmínkách, v cyklech nebo ve výpočtech.[17]

4.5 Podmínky

Podmínky v Pythonu se používají k řízení toku programu na základě určitých podmínek. Pomocí podmínek můžeme rozhodovat, který kód se má vykonat a který se má přeskočit, v závislosti na splnění určitého logického výrazu. V Pythonu se používá klíčové slovo if k definici podmínky. Podmínka je vyhodnocena jako buď pravdivá (*True*) nebo nepravdivá (*False*). Pokud je podmínka pravdivá, vykoná se blok kódu přiřazený ke klíčovému slovu if. Pokud je podmínka nepravdivá, blok kódu se přeskočí nebo se provede alternativní blok kódu přiřazený klíčovému slovu *else* nebo *elif* (zkratka pro „*else if*“).[18]

```
if(podminka):
    "blok kódu, který se provede zdali podmínka true"
elif(jina_podminka):
    "blok kódu, který se provede zdali první podmínka bude false a jina_podminka bude true"
else:
    "blok kódu, který se provede zdali předešlé podmínky budou false"
```

Obrázek 10: Ukázka podmínky if,elif,else

4.6 Cykly

V Pythonu existují různé typy cyklů, které umožňují opakovat určitý blok kódu až do splnění určité podmínky. Tímto způsobem můžeme provádět opakované akce nebo iterovat přes kolekce dat.

4.6.1 Cyklus while

Cyklus *while* v Pythonu se používá k opakování určitého bloku kódu, dokud je splněna určitá podmínka. Blok kódu se vykonává opakovaně, dokud je podmínka pravdivá. Jakmile je podmínka nepravdivá, cyklus se ukončí a program pokračuje dál. Nejprve se vyhodnotí podmínka. Pokud je podmínka pravdivá, provede se blok kódu uvnitř cyklu. Poté se podmínka znovu vyhodnotí. Tento proces se opakuje, dokud je podmínka pravdivá. Jakmile je podmínka nepravdivá, cyklus se ukončí a program pokračuje dál po bloku *while*. Je důležité zajistit, aby podmínka cyklu *while* byla nakonec někdy nepravdivá, aby se zabránilo nekonečnému opakování cyklu (zacyklení).[19]

```
while(podminka):
    "blok kódu, který se bude opakovat dokud bude podmínka true"
```

Obrázek 11: Ukázka cyklu while

4.6.2 Cyklus for

Cyklus *for* je v Pythonu používán k procházení prvků v iterovatelné posloupnosti, jako je seznam, n-tice, řetězec nebo slovník. Syntaxe cyklu *for* je následující:

```
for prvek in posloupnost:
    "blok kódu, který se bude opakovat do konce posloupnosti"
```

Obrázek 12: Ukázka cyklu for

Cyklus *for* postupně při každém opakování přiřazuje aktuální prvek z posloupnosti proměnné prvek a provede blok kódu uvnitř cyklu. Po dokončení bloku kódu se cyklus *for* posune na další prvek v posloupnosti a pokračuje v dalším opakování. Tento proces pokračuje, dokud nejsou zpracovány všechny prvky v posloupnosti. [19]

5 OBJEKOTOVĚ ORIENTOVANÉ PROGRAMOVÁNÍ V JAZYCE PYTHON

5.1 Základní pojmy OOP

Objektově orientované programování (OOP) je významnou součástí Pythonu a poskytuje programátorům více strukturovaný a modulární způsob psaní kódu. V OOP jsou všechny prvky programu (proměnné, funkce atd.) organizovány do tzv. objektů. Tyto objekty mají určité vlastnosti, které jsou popsány pomocí tříd. Třídy jsou vlastně šablony, které definují vlastnosti a chování objektů. Vytvářením objektů z tříd se vytváří instance objektů, které mohou mít odlišné hodnoty vlastností, ale vždy mají stejný základní typ. Veškeré vlastnosti a metody objektů jsou k nim přiřazeny pomocí speciálního atributu „*self*“, který se při každém volání metody předává automaticky a umožňuje tak přístup k danému objektu. Hlavními výhodami OOP jsou znovupoužitelnost kódu, lepší organizace kódu, větší flexibilita a zvýšená bezpečnost. Nicméně pro začátečníky může být koncept OOP složitý, a proto se doporučuje nejdříve se naučit základy programování a až poté se pustit do OOP.

5.2 Třídy

Třídy jsou základním stavebním prvkem objektově orientovaného programování v Pythonu. Třída je definice objektového typu, která slouží jako šablona pro vytváření konkrétních instancí objektů. Každá třída může obsahovat atributy (data) a metody (funkce), které definují chování a vlastnosti objektů vytvořených z této třídy. Vytvoření třídy v Pythonu začíná klíčovým slovem *class*, následovaným názvem třídy. Konvence pojmenování tříd je použít CamelCase notaci, kde každé slovo začíná velkým písmenem. Příklad definice třídy:

Obrázek 13: Ukázka definice třídy

V tomto příkladu je definována třída Person, která má atributy *name* a metodu *greet()*. Metoda *__init__()* je speciální metoda, která se volá při vytváření nové instance třídy. Pomocí *self* se odkazujeme na aktuální instanci objektu.

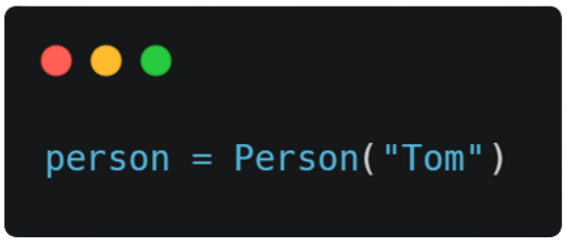
Třídy umožňují vytvářet objekty, které sdílejí společné vlastnosti a chování. Tímto způsobem můžeme organizovat a strukturovat náš kód, což usnadňuje jeho správu a rozšiřitelnost.[20]

5.3 Objekty

V Pythonu je všechno objekt. Objekt je konkrétní instance třídy, která obsahuje data a metody. Každý objekt má svůj stav (data) a chování (metody), které jsou definovány třídou, ze které byl vytvořen. Objekty v Pythonu mají několik klíčových vlastností:

- Identita: Každý objekt má unikátní identitu, která je přiřazena při jeho vytvoření. Identita objektu se nemění po celou dobu životnosti objektu a můžeme ji získat pomocí funkce *id()*.
- Stav: Stav objektu je určen jeho atributy (data). Atributy jsou proměnné připojené k objektu a uchovávají informace o jeho stavu. K atributům můžeme přistupovat pomocí tečkové notace.
- Chování: Chování objektu je definováno jeho metodami (funkcemi), které jsou připojeny k objektu a umožňují provádět různé akce. Metody objektu mohou manipulovat s jeho atributy nebo provádět další operace.

Vytvoření objektu v Pythonu se provádí pomocí instancování třídy. Používáme syntaxi „objekt = Třída()“, kde objekt je proměnná, do které uložíme nově vytvořený objekt, a Třída je název třídy, ze které objekt vytváříme. Příklad vytvoření objektu z předešlého obrázku „Ukázka definice třídy“:



Obrázek 14: Ukázka vytvoření objektu

V tomto příkladu jsme vytvořili objekt person z třídy Person. Objekt má atribut *name* nastavený na „Tom“. Objekty vytvořené z tříd mohou volat metody a přistupovat k atributům pomocí tečkové notace:

Obrázek 15: Ukázka volání metody

Objekty v Pythonu nám umožňují pracovat s daty a funkcemi jako jednotný celek, což nám usnadňuje organizaci a modularitu našeho kódu.[21]

5.4 Dědičnost

Dědičnost je důležitým konceptem objektově orientovaného programování, který umožňuje vytvářet hierarchie tříd a sdílet jejich vlastnosti a chování. V jazyce Python je dědičnost implementována pomocí klíčového slova „class“. Třída, která dědí vlastnosti a metody z jiné třídy, se nazývá potomkem (nebo také podtřídou), zatímco třída, ze které jsou děděny vlastnosti, se nazývá rodičovská třída (nebo také nadřízená třída). Potomek může zdědit veškeré vlastnosti (atributy) a metody rodičovské třídy a může je dále rozšiřovat nebo přepisovat podle svých potřeb. Dědičnost umožňuje efektivní sdílení kódu a snižuje opakované psaní stejného kódu. Potomci mohou přebírat obecné vlastnosti a chování rodičovské třídy a dále je specializovat nebo rozšiřovat o nové vlastnosti a metody. V Pythonu se dědičnost zapisuje ve tvaru:

Obrázek 16: Ukázka dědičnosti

Tímto zápisem *ChildClass* zdědí veškeré vlastnosti a metody z *ParentClass*. Poté může *ChildClass* přidávat další vlastnosti a metody nebo přepisovat existující. Dědičnost v Pythonu umožňuje vytvářet hierarchie tříd a organizovat kód do logických celků. Je to mocný nástroj pro strukturování a modularitu programů.[20]

5.5 Polymorfismus

Polymorfismus je důležitým konceptem objektově orientovaného programování, který umožňuje pracovat s objekty různých tříd jako s jednotným rozhraním. V jazyce Python je polymorfismus zajištěn díky dynamickému typování. Polymorfismus v Pythonu umožňuje, aby různé objekty reagovaly na stejné volání metody, ačkoli mohou mít různou implementaci této metody. To znamená, že objekty stejného základního typu (rodičovské třídy) mohou vykonávat různé akce nebo poskytovat různé výsledky při volání stejné metody. Jedním ze způsobů, jak dosáhnout polymorfismu v Pythonu, je použití dědičnosti a přepsání (*override*) metod ve třídách potomků. Potomek může přepsat (změnit implementaci) metody své rodičovské třídy a poskytnout tak specifické chování. Dalším způsobem je využití konceptu rozhraní. Python sice nemá striktní podporu rozhraní jako jiné jazyky, ale pomocí duck typingu umožňuje pracovat s objekty, které vyhovují určitému rozhraní (mají příslušné metody) bez ohledu na jejich konkrétní třídu. Polymorfismus v Pythonu přináší flexibilitu a znuvupoužitelnost kódu. Díky tomu můžeme pracovat s různými objekty jednotným způsobem a snadno rozšiřovat nebo měnit chování programu. [20]

5.6 Abstraktní třídy

V jazyce Python je možné definovat abstraktní třídy, které slouží jako šablony pro odvození konkrétních tříd. Abstraktní třída je třída, která obsahuje jednu nebo více abstraktních metod. Abstraktní metody jsou deklarovány, ale nemají implementaci. Slouží pouze k definici rozhraní a předepsání, jaké metody musí být v odvozených třídách implementovány. Abstraktní třída sama o sobě nemůže být instancovaná, ale může být použita k odvození konkrétních tříd, které implementují abstraktní metody. Odvozené třídy musí implementovat všechny abstraktní metody definované v abstraktní třídě, jinak budou považovány za abstraktní třídy samy o sobě. Pomocí abstraktních tříd a metod je možné definovat společné rozhraní pro skupinu tříd a zajistit, že tyto třídy implementují určité chování. Abstraktní třídy slouží jako kontrakt pro odvozené třídy a pomáhají zajistit konzistenci a strukturu v kódu. V jazyce Python je pro definici abstraktních tříd a metod k dispozici modul abc

(Abstract Base Classes), který poskytuje funkce pro vytváření abstraktních tříd a metod. Tento modul umožňuje definovat abstraktní třídy pomocí dekoratéru @abstractmethod a vynucuje, aby odvozené třídy implementovaly abstraktní metody. [22]

6 KNIHOVNY JAZYKA PYTHON

Knihovny jazyka Python jsou kolekce kódů napsaných v jazyce Python, které obsahují funkce, metody a třídy, které rozšiřují základní funkčnosti jazyka Python. Knihovny mohou být využity pro různé účely, jako například matematické výpočty, práci s daty, tvorbu webových aplikací, práci s grafikou a mnoho dalších. Díky knihovnám lze výrazně zkrátit čas potřebný pro vývoj aplikace a zlepšit její výkon. Knihovny jsou často součástí standardní distribuce jazyka Python, ale mohou být také instalovány z externích zdrojů, jako jsou repozitáře PyPI (Python Package Index) nebo GitHub. Tyto knihovny jsou často vyvíjeny a udržovány komunitou Pythonu, což zajišťuje jejich kvalitu a aktualizace. Použití knihoven je velmi důležité pro efektivní vývoj aplikací v Pythonu. Knihovny zajišťují širokou funkcionalitu pro programátory a poskytují různé nástroje pro řešení problémů v oboru, pro který jsou určeny. Například knihovna NumPy se zaměřuje na matematické výpočty, knihovna Pandas na práci s daty a knihovna Flask na tvorbu webových aplikací. Knihovny jsou tedy velmi užitečné a důležité pro práci v jazyce Python, proto je důležité naučit se je používat a vybírat ty nejvhodnější pro danou úlohu.

6.1 Nejčastěji používané moduly

Jazyk Python je oblíbený díky širokému spektru knihoven, které jsou k dispozici. Tyto knihovny zjednodušují vývoj aplikací a programů tím, že poskytují předem napsaný kód, který řeší specifické problémy. Následující jsou některé z nejčastěji používaných knihoven jazyka Python:

- NumPy – knihovna pro vědecké výpočty s využitím matic a vektorů.
- Pandas – knihovna pro analýzu a manipulaci s daty, včetně funkcí pro načítání a zápis dat z různých zdrojů.
- Matplotlib – knihovna pro vizualizaci dat, umožňuje tvorbu různých typů grafů.
- Scikit-learn – knihovna pro strojové učení, poskytuje řadu algoritmů pro klasifikaci, regresi a shlukování dat.
- TensorFlow – knihovna pro strojové učení, zaměřená na tvorbu a trénování neuronových sítí.
- Django – knihovna pro tvorbu webových aplikací, poskytuje nástroje pro tvorbu uživatelského rozhraní a komunikaci s databází.

- Flask – lehká knihovna pro tvorbu webových aplikací, která je méně strukturovaná než Django, ale zároveň poskytuje dostatečnou flexibilitu.

Tato seznam je samozřejmě pouze malou ukázkou z množství dostupných knihoven a každá knihovna má své specifické použití v závislosti na potřebách vývojářů.[23, 24]

6.2 Použití knihoven NumPy, Pandas a Matplotlib

6.2.1 Numpy

Knihovna NumPy (Numerical Python) je jedna z nejvýznamnějších a nejčastěji používaných knihoven v jazyce Python pro vědecké výpočty. NumPy poskytuje podporu pro práci s maticemi, vektorovými operacemi, rychlými lineárními algoritmy, matematickými funkcemi, náhodnými čísly a dalšími funkcemi potřebnými pro numerické výpočty. Jedním z největších přínosů knihovny NumPy je možnost využití mnoha funkcí pro vektorové a matematické operace na velkých datasetech. Tyto operace jsou velmi efektivní a rychlé, což zajišťuje, že programy budou rychleji a efektivněji zpracovávat data. Knihovna NumPy je také často používána v kombinaci s dalšími knihovnami, jako je například knihovna Matplotlib pro vizualizaci dat. NumPy tedy hraje klíčovou roli v ekosystému vědeckých výpočtů v jazyce Python.[25]

6.2.2 Pandas

Python knihovna Pandas je nástroj pro manipulaci s daty a analýzu dat, který umožňuje snadno a rychle načítat, ukládat a manipulovat s daty v různých formátech, včetně CSV, Excelu a SQL databází. Pandas nabízí širokou škálu funkcí pro přehledné zpracování a analýzu dat, jako jsou agregace, spojování, skupinování a filtrování dat. Tato knihovna je často používána v oblasti strojového učení, financí, biologie a dalších vědních oblastech. Pandas v sobě zahrnuje dvě základní struktury dat: *DataFrame* a *Series*. *DataFrame* je dvourozměrná tabulka, která obsahuje řádky a sloupce, kde každý sloupec může mít odlišný datový typ. Series je jednorozměrná struktura dat, která obsahuje řadu hodnot s indexem. Pandas také obsahuje mnoho nástrojů pro zpracování chybějících dat, úpravu datových typů, přejmenování sloupců a mnoho dalšího. Knihovna Pandas výrazně zjednodušuje manipulaci s daty a umožňuje uživatelům rychle a efektivně analyzovat velké soubory dat.[26, 27]

6.2.3 Matplotlib

Knihovna Matplotlib je jednou z nejrozšířenějších knihoven pro vizualizaci dat v jazyce Python. Tato knihovna poskytuje uživatelům vysokou úroveň flexibility a možností přizpůsobení. S Matplotlibem je možné vytvářet mnoho druhů grafů, včetně čar, sloupcových, histogramů, scatter plotů a mnoho dalších. Kromě toho umožňuje Matplotlib i interaktivní vizualizaci a práci s animacemi. Hlavní prvek v Matplotlibu je obvykle graf, který se skládá z jednoho nebo více podgrafů (tzv. subplotů). Každý *subplot* může mít svou vlastní osu X a Y, popisky os, titulky a mnoho dalšího. Matplotlib poskytuje také mnoho způsobů, jak přizpůsobit vzhled grafů, včetně změny barvy, stylu a velikosti čar, použití různých typů markerů a mnoho dalšího. Matplotlib je jednou z nejčastěji používaných knihoven pro vizualizaci dat v jazyce Python a je součástí mnoha dalších knihoven, jako je například Pandas. Knihovna Matplotlib je k dispozici zdarma a je vyvíjena pod open source licencí.[28]

II. PRAKTICKÁ ČÁST

7 VÝUKOVÉ BALÍKY

Výukové balíky byly vytvořeny za účelem výuky základů programovacího jazyka Python. Obsahují teorii, ukázky a příklady na procvičení dané látky.

7.1 Úvod

První PowerPoint výukový balík nazývající se „Python – úvod“ se zaměřuje na základní informace o jazyce Python a jeho využití. Prezentace začíná definicí jazyka Python a jeho historií, včetně krátkého přehledu jeho vývoje a vývojářského prostředí. Následuje část o využití Pythonu, kde jsou popsány oblasti, ve kterých je Python nejčastěji používán, jako například vědecké výpočty, strojové učení nebo webová aplikace. Další část se věnuje vlastnostem jazyka Python, jako jsou jednoduchost syntaxe, modularita a kompatibilita s množstvím knihoven, a také důvodům, proč je Python často považován za výhodný volbu pro programátory. Poté se prezentace soustředí na praktické záležitosti, jako je instalace Pythonu a vývojového prostředí. V této části jsou krokové instrukce k instalaci Pythonu a VS Code, populárního kódu editoru, a přípravě prostředí pro psaní a spouštění Python souborů. Následuje praktická ukázka, kde jsou uvedeny kroky, jak vytvořit nový Python soubor a napsat v něm program „Hello World“.

7.2 Syntaxe

Druhý PowerPoint výukový balík „Python – syntaxe“ se zaměřuje na základní syntaxi jazyka Python. V první části je uveden úvod do syntaxe jazyka Python a jak se liší od syntaxe jiných programovacích jazyků. Poté se věnuje konkrétním pravidlům syntaxe. Ukazuje se, jakým způsobem se v Pythonu používá odsazení k označení bloků kódu, což je výrazný rozdíl oproti jiným jazykům. Dále se vysvětluje používání středníků a komentářů, aby se kód stal čitelnější a snadněji udržovatelný. Další část se věnuje klíčovým slovům v Pythonu. Ty jsou zásadní pro vytváření kódu, protože určují, jak bude kód interpretován. V této části jsou popsána nejčastěji používaná klíčová slova, jako jsou „*if*“, „*else*“, „*for*“ a „*while*“. Nakonec se věnuje proměnným v Pythonu. Popisuje se, jakým způsobem se deklarují a jakým způsobem se jim přiřazuje hodnota. Cílem tohoto výukového balíku je naučit úplné začátečníky základní syntaxi jazyka Python, aby mohli začít psát jednoduché programy a postupně se dále rozvíjet.

7.3 Datové typy

Třetí PowerPoint výukový balík s názvem „Python – datové typy“ se zaměřuje na základní datové typy v jazyce Python. Prezentace začíná úvodem do datových typů a jejich významu v programování. Poté se přechází k číselným datovým typům, jako jsou celá čísla (*integers*), desetinná čísla (*floats*) a komplexní čísla (*complex numbers*). Jsou popsány jejich vlastnosti a způsoby práce s nimi v Pythonu. Následuje část věnovaná řetězcům (*strings*), což jsou posloupnosti znaků. Následuje stručná sekce o sekvenčních typech. Prezentace končí sekci o přetypování, což je proces konverze jednoho datového typu na jiný.

7.4 Podmínky

Čtvrtý výukový balík s názvem „Python – podmínky“ se zaměřuje na podmínky v jazyce Python. Po úvodu se věnuje relačním a logickým operátorům a vysvětluje, jakým způsobem lze porovnávat hodnoty v Pythonu a jaké logické operátory se používají pro spojování podmínek. Následně se balík věnuje jednotlivým typům podmínek, začínaje jednoduchou *If* podmínkou, která umožňuje vykonat určitý kód pouze v případě, že je splněna určitá podmínka. Dále se pak vysvětluje použití *Elif* a *Else* podmínek, které rozšiřují možnosti vykonání kódu v závislosti na splnění různých podmínek. Další kapitola se pak zaměřuje na vnořené podmínky, což je způsob, jakým lze vnořit jednu podmínku do druhé. Tento postup se využívá zejména v případě, kdy je potřeba vyhodnotit více podmínek současně. Poslední kapitola tohoto výukového balíku se věnuje krátkému zápisu podmínek, což je zkrácená forma *If* podmínky, která umožňuje v jediné řádce zapsat podmínku a vykonat kód, pokud je podmínka splněna. Tento způsob zápisu je užitečný zejména pro jednoduché podmínky, které nevyžadují větší rozsah kódu.

7.5 Cykly

Tento pátý výukový balík se zaměřuje na Python cykly. V úvodu je stručně vysvětleno, jak fungují cykly v Pythonu a proč je důležité je používat. Poté jsou představeny dva hlavní typy cyklů, *for* a *while* cykly. *For* cyklus je vhodný pro opakování kódu na pevně stanovený počet iterací, zatímco *while* cyklus je používán pro opakování kódu, dokud je splněna určitá podmínka. V rámci prezentace jsou také ukázány příklady použití těchto cyklů a jejich syntaxe. Dále jsou v prezentaci představeny vnořené cykly, což jsou cykly, které jsou umístěny v rámci jiného cyklu. Tyto cykly jsou užitečné pro opakování kódu v rámci složitějších programů. V prezentaci jsou také popsány dvě důležité instrukce, které mohou

být použity v rámci cyklů. První instrukce je *continue*, která slouží k přeskočení aktuální iterace cyklu a přejití k další iteraci. Druhá instrukce je *break*, která zastaví cyklus, jakmile je splněna určitá podmínka. Prezentace také vysvětluje, jakým způsobem může být použití cyklů zlepšeno pomocí funkcionality Pythonu, jako je funkce *range()*.

7.6 Řetězce

V šestém PowerPoint výukovém balíku s názvem „Python – řetězce“ se budeme věnovat tématům týkajícím se práce s řetězci v jazyce Python. Začneme úvodem, kde se dozvíme, co jsou řetězce v Pythonu a jak je vytváříme. Dále se podíváme na speciální znaky v řetězcích, jako například zpětné lomítko n (\n), a jak je použít. V další části se naučíme, jak pracovat s řetězci, jak je spojovat, opakovat, vybírat určité části a podobně. Poté se podíváme na hledání v řetězcích pomocí různých metod a operátorů. Například se naučíme, jak zjistit délku řetězce, najít konkrétní podřetězec nebo zjistit, zda se určitý podřetězec v řetězci vyskytuje. Na závěr se podíváme na formátování řetězců, což je velmi užitečné pro přehledné vypsání dat. Naučíme se používat speciální formátovací řetězce, které nám umožní vkládat proměnné a konstanty do textu a upravovat jejich výstupní formát. Celkově tedy tento výukový balík poskytuje ucelený pohled na práci s řetězci v jazyce Python, včetně základních metod a funkcí pro manipulaci s nimi.

7.7 Sekvenční typy

V sedmém PowerPoint výukovém balíku s názvem „Python – sekvenční typy“ se budeme zaměřovat na různé druhy sekvenčních typů v jazyce Python. Začneme úvodem, kde se seznámíme s konceptem sekvenčních typů a jejich využitím v programování. Poté se budeme věnovat prvnímu typu – listům. Nejprve si zopakujeme, co jsou listy a jak je vytváříme. Následně se naučíme různé operace a manipulace s listy, jako je přidávání, odebírání a upravování prvků, třídění, vyhledávání a indexování. Dalším typem, který si probereme, jsou tuply. Opět si zopakujeme jejich vytváření a následně se naučíme, jak s nimi pracovat. Budeme se zabývat jejich neměnností a jak využít různé operace a metody k práci s tuply. Dále se zaměříme na sety. Opět si zopakujeme jejich vytváření a poté se naučíme různé operace a metody pro práci se sety. Budeme se zabývat sjednocením, průnikem a rozdílem mezi sety, jejich přidáváním a odebíráním prvků a dalšími operacemi. Posledním sekvenčním typem, který probereme, jsou slovníky. Zopakujeme si jejich vytváření a poté se budeme věnovat různým operacím a metodám, které nám umožní pracovat se slovníky. Nau-

číme se přidávat a odebírat klíče a hodnoty, vyhledávat a upravovat hodnoty a další. Cílem tohoto výukového balíku je poskytnout úplný přehled o různých sekvenčních typech v Pythonu a naučit se efektivně s nimi pracovat.

7.8 Funkce

V osmém PowerPoint výukovém balíku s názvem „Python – funkce“ se budeme zabývat funkcemi, které jsou základním stavebním kamenem programování. Začneme úvodem, kde se seznámíme se základními koncepty funkcí a jejich využitím v Pythonu. Poté se budeme věnovat syntaxi funkcí. Naučíme se, jak definovat funkce a jakým způsobem uvádět parametry. Probereme různé způsoby pojmenování funkcí a význam jednotlivých částí definice funkce. Dále se budeme zaměřovat na volání funkcí. Naučíme se, jak volat funkce a jak předávat argumenty. Probereme povinné argumenty, které musí být předány do funkce, a také nepovinné argumenty, které mají výchozí hodnotu a mohou být vynechány. Budeme se také věnovat keyword argumentům, což je způsob předávání argumentů do funkce pomocí jejich jména. Naučíme se využívat tuto funkcionalitu pro přehlednější a flexibilnější práci s funkcemi. Dalším tématem jsou libovolný počet argumentů a libovolný počet keyword argumentů. Naučíme se definovat funkce, které mohou přijímat proměnlivý počet argumentů a keyword argumentů. Bude zde také diskutováno, jak s těmito argumenty pracovat uvnitř funkce. Nakonec se budeme věnovat příkazu return, který umožňuje funkci vrátit hodnotu zpět volajícímu kódu. Probereme syntaxi a význam výrazu *return*. Na závěr se budeme krátce zabývat rekurzí, což je technika, při které funkce volá samu sebe. Probereme základní principy rekurze a její využití v Pythonu. Cílem tohoto výukového balíku je poskytnout ucelený přehled o funkcích v Pythonu a naučit se efektivně je využívat pro strukturování a opakované využití kódu.

7.9 Moduly

V devátém PowerPoint výukovém balíku s názvem „Python – moduly“ se budeme věnovat práci s moduly v jazyce Python. Začneme úvodem, kde si vysvětlíme, co jsou moduly a proč jsou důležité při organizaci a strukturování kódu. Následovat bude seznámení se vestavěnými moduly v Pythonu. Uvedeme si i často používané vestavěné moduly. Dále se budeme věnovat instalaci externích modulů. Ukážeme si, jak pomocí nástroje pip instalovat moduly z Python Package Index (PyPI), což je rozsáhlý repozitář dostupných modulů. Poté se zaměříme na importování modulů do našeho programu. Vysvětlíme různé způsoby

importování, včetně importování celého modulu, importování konkrétních funkcí nebo tříd z modulu. Následně se budeme věnovat tvorbě vlastních modulů. Ukážeme si, jak vytvořit vlastní modul obsahující funkce nebo proměnné, a jak tento modul importovat do jiných programů. Na závěr se budeme zabývat použitím aliasů pro moduly. Vysvětlíme, jak pomocí aliasů můžeme měnit názvy modulů při importování, což může usnadnit práci s dlouhými nebo složitými názvy modulů. Cílem tohoto výukového balíku je seznámit se s konceptem modulů v Pythonu, naučit se importovat vestavěné i externí moduly, vytvářet vlastní moduly a pracovat s aliasy pro zkrácení názvů modulů.

7.10 Práce se soubory

V desátém PowerPoint výukovém balíku s názvem „Python – práce se soubory“ se budeme zaměřovat na manipulaci se soubory v jazyce Python. Představíme si různé operace, které umožňují otevírat, číst, zapisovat a mazat soubory. Začneme sekcí otevírání a zavírání souborů, kde se naučíme, jak otevřít soubor pomocí vestavěné funkce *open()* a jakým způsobem zavřít soubor po dokončení práce. Ukážeme si také, jak nastavit různé režimy otevření souboru pro čtení („r“), zápis („w“), přidání („a“) nebo binární mód („b“). Dále se zaměříme na čtení ze souborů, kde ukážeme různé způsoby čtení dat ze souboru. Budeme se zabývat čtením celého obsahu souboru, čtením řádek nebo čtením po blocích dat. Vysvětlíme také použití metod pro postupné čtení jednotlivých řádek ze souboru. Poté se budeme věnovat zápisu do souboru. Naučíme se, jak zapisovat data do souboru pomocí režimu zápisu („w“) a také jak přidávat data na konec souboru pomocí režimu přidání („a“). Ukážeme si také, jak pracovat s formátováním dat před jejich zápisem do souboru. Následně se budeme zabývat vytvářením nových souborů. Vysvětlíme, jak vytvořit nový soubor pomocí metody *open()* s režimem zápisu. Na závěr se budeme věnovat mazání souborů, kde si ukážeme, jak pomocí vestavěného modulu („os“) a její funkcí *remove()* odstranit soubor z disku. Vysvětlíme také důležitost opatrnosti při používání této operace. Cílem tohoto výukového balíku je seznámit se s práci se soubory v jazyce Python. Naučíme se otevírat a zavírat soubory, číst z nich data, zapisovat do nich a také vytvářet a mazat soubory.

7.11 Matplotlib

V jedenáctém PowerPoint výukovém balíku s názvem „Python – Matplotlib“ se budeme zaměřovat na knihovnu Matplotlib, která slouží k vizualizaci dat a tvorbě grafů v jazyce

Python. Začneme sekcí o úvodu do Matplotlibu, kde se seznámíme s hlavními vlastnostmi knihovny a jejím využitím. Ukážeme si také, jak nainstalovat knihovnu pomocí nástroje pip a jak ji importovat do našeho Python projektu. Dále se budeme věnovat základním typům grafů, které můžeme vytvářet pomocí Matplotlibu. Představíme si čárový graf, sloupcový graf a koláčový graf. Vysvětlíme, jak připravit data pro jednotlivé typy grafů a jak je vykreslit pomocí funkcí poskytovaných knihovnou. Další část se bude zabývat úpravou vzhledu grafů. Naučíme se, jak měnit barvy, šířky čar, velikost markerů a další parametry grafů. Ukážeme si také, jak přidávat popisky os, nadpisy a legendy do grafu. Nakonec se budeme věnovat vykreslování více grafů na jednom obrázku. Představíme si možnosti rozdělení jednoho obrázku na různé panely a umístění grafů na těchto panelech. Cílem tohoto výukového balíku je seznámit se s knihovnou Matplotlib a naučit se vytvářet základní typy grafů. Naučíme se upravovat vzhled grafů a také vykreslovat více grafů na jednom obrázku. Tímto získáme schopnost vizualizovat data a prezentovat je prostřednictvím grafů a diagramů.

7.12 Numpy

Dvanáctý PowerPoint výukový balík s názvem „Python – NumPy“ se zaměřuje na knihovnu NumPy, která je základem pro efektivní práci s numerickými daty v jazyce Python. V úvodní části se seznámíme s NumPy a jeho hlavními vlastnostmi. Vysvětlíme, proč je NumPy tak důležitý pro vědecké výpočty a manipulaci s daty. Ukážeme si také, jak nainstalovat NumPy pomocí nástroje pip a jak jej importovat do našeho Python projektu. Poté se budeme věnovat vytváření polí, což je základní datová struktura v NumPy. Představíme si různé funkce pro vytváření polí, například *array()*, *arange()* a *linspace()*. Vysvětlíme, jakým způsobem můžeme specifikovat rozměry a hodnoty polí. Dále se budeme zabývat indexováním a přístupem k hodnotám v polích. Vysvětlíme, jakým způsobem můžeme přistupovat k jednotlivým prvkům pole a jak indexovat vícedimenzionální pole. Budeme také diskutovat o konceptu *copy* a *view*, který je důležitý pro správnou manipulaci s poli v NumPy. Vysvětlíme rozdíl mezi vytvářením kopie a vytvořením pohledu na původní pole. Následně se zaměříme na pokročilé indexování, které nám umožní získat kopii pole. Nakonec se budeme věnovat univerzálním funkcím (*ufuncs*) v NumPy. Tyto funkce umožňují provádět různé matematické operace a manipulace s poli efektivněji než standardní Python funkce. Cílem tohoto výukového balíku je seznámit se s knihovnou NumPy a naučit se vytvářet a manipulovat s poli. Budeme se také zabývat indexováním, kopírováním a po-

kročilými technikami práce s poli. NumPy je klíčovou knihovnou pro práci s daty v Pythonu a poskytuje nám výkonné nástroje pro numerické výpočty a analýzu dat.

7.13 Pandas

Třináctý PowerPoint výukový balík s názvem „Python – Pandas“ se zaměřuje na knihovnu Pandas, která je výkonným nástrojem pro analýzu a manipulaci s daty v jazyce Python. Na začátku výukového balíku se seznámíme s knihovnou Pandas a jejími hlavními vlastnostmi. Vysvětlíme, proč je Pandas tak populární pro práci s daty a jak jej nainstalovat pomocí nástroje pip. Poté si ukážeme, jak Pandas importovat do našeho Python projektu. Dále se budeme věnovat datovým strukturám v Pandas. První datovou strukturou, se kterou se seznámíme, je *Series*. Vysvětlíme, jak vytvářet a manipulovat s objekty *Series*. Naučíme se také, jak označovat řádky v objektu *Series* a jak přistupovat k hodnotám. Dalším důležitým konceptem v Pandas jsou *DataFrames*. Vysvětlíme, jak vytvářet *DataFrames* a jakým způsobem se jednotlivé sloupce nazývají a indexují. Naučíme se také přistupovat k hodnotám v *DataFrames* a jak odebírat a vkládat hodnoty do tabulky. Cílem tohoto výukového balíku je seznámit se s knihovnou Pandas a naučit se pracovat s datovými strukturami, jako jsou *Series* a *DataFrames*. Pandas je mocným nástrojem pro analýzu a manipulaci s daty, který nám umožňuje provádět různé operace, jako je filtrování, agregace a transformace dat.

7.14 Jupyter

Čtrnáctý PowerPoint výukový balík s názvem „Python – Jupyter“ se zaměřuje na praktické používání nástroje Jupyter v prostředí VS Code. Balík obsahuje informace o základních konceptech Jupyteru a postupu jejich využití. Úvodní slide představuje význam a využití Jupyteru v programování a datové analýze. Důraz je kladen na výhody, které Jupyter poskytuje při práci s interaktivními notebooky. Následující slide se zaměřuje na instalaci Jupyteru do prostředí VS Code. Je zde představeno rozšíření Jupyter pro VS Code a poskytnut postup pro správnou instalaci a konfiguraci prostředí, aby byl Jupyter plně funkční a připraven k použití. Další slide s názvem "Vytvoření notebooku" popisuje proces vytváření nového Jupyter notebooku v prostředí VS Code. Zahrnuje volbu názvu a umístění notebooku a představuje základní rozhraní Jupyter notebooku. Slide s názvem „Buňky“ se zaměřuje na koncept buněk v Jupyteru. Vysvětluje rozdělení buněk na textové a kódové. Další slide s názvem „Přidávání buněk“ představuje postup přidání nových buněk do notebooku.

Umožňuje uživatelům rozšiřovat svůj notebook o další obsah a upravovat ho podle svých potřeb. Dále si ukážeme, jak můžeme spouštět kód buňkách. Vysvětluje, jak vykonávat kód v buňkách a zobrazovat výsledky. Poslední slide s názvem „Odstranění buněk“ ukazuje, jak odstraňovat nepotřebné nebo chybné buňky z notebooku. Uživatelé se dozvědí, jak správně odstranit buňky a udržovat svůj notebook čistý a přehledný. Tento výukový balík „Python – Jupyter“ poskytuje užitečné informace pro začínající uživatele Jupyteru, aby mohli efektivně využívat tento interaktivní nástroj v prostředí VS Code.

8 VÝUKOVÉ VIDEA

Výuková videa, která využívají čtrnácti předešle zmíněných výukových balíků, přinášejí uživatelům možnost audiovizuálního vzdělávání v oblasti Pythonu. Tato videa jsou zaměřena na různé aspekty Pythonu, jeho knihoven a poskytují uživatelům přehledně strukturované výklady jednotlivých témat. Každé výukové video se zaměřuje na konkrétní výukový balík a představuje uživatelům jeho základní koncepty, syntaxi, použití a praktické příklady. Videa jsou navržena tak, aby byla snadno srozumitelná pro začátečníky i pokročilejší uživatele. Každé video se snaží prezentovat téma srozumitelným a poutavým způsobem, často doplněné příklady, které umožňují uživatelům lépe porozumět danému konceptu a aplikovat jej ve vlastní praxi.

ZÁVĚR

V rámci této bakalářské práce bylo provedeno zkoumání a analýza problematiky výuky na vysoké škole. Na základě tohoto výzkumu bylo navrženo a implementováno čtrnáct výukových balíků a videí. Každý z těchto výukových balíků se zaměřuje na výuku Pythonu a poskytuje uživatelům srozumitelný a strukturovaný přehled daného tématu. Balíky jsou navrženy tak, aby uživatelé mohli postupovat od základů a postupně se rozvíjet ve svých znalostech Pythonu.

V rámci praktické části práce byly také vytvořeny výuková videa, která převádějí jednotlivé výukové balíky do audiovizuální formy. Tato videa umožňují uživatelům získat teoretické zkušenosti s Pythonem a jeho knihovnami prostřednictvím výkladu a ukázek.

Cílem této práce bylo poskytnout studentům a zájemcům o Python přístupný a efektivní způsob výuky. Vytvořené výukové balíky a videa by měli sloužit jako cenný nástroj pro rozšíření dovedností v oblasti programování v Pythonu. Práce přináší přínos nejen studentům, ale i učitelům, kteří mohou využít vytvořené materiály pro svou výuku a prezentace. Tato práce by měla přispět k rozvoji výuky Pythonu a pomoct širokému spektru uživatelů při osvojování tohoto populárního programovacího jazyka.

SEZNAM POUŽITÉ LITERATURY

[1] What is Python? Executive Summary. *Python.org* [online]. [vid. 2023-05-19]. Dostupné z: https://www.python.org/doc/essays/blurb/

[2] A Brief History of Python. *LearnPython.com* [online]. 20. červen 2022 [vid. 2023-05-19]. Dostupné z: https://learnpython.com/blog/history-of-python/

[3] What Is Python Used For? A Beginner's Guide. *Coursera* [online]. 17. květen 2023 [vid. 2023-05-19]. Dostupné z: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

[4] What is an IDE? - Integrated Development Environment Explained - AWS. *Amazon Web Services, Inc.* [online]. [vid. 2023-05-19]. Dostupné z: https://aws.amazon.com/what-is/ide/

[5] IDLE. *Python documentation* [online]. [vid. 2023-05-19]. Dostupné z: https://docs.python.org/3/library/idle.html

[6] *Setting Up VSCode For Python: A Complete Guide* [online]. [vid. 2023-05-19]. Dostupné z: https://www.datacamp.com/tutorial/setting-up-vscode-python

[7] Features - PyCharm. *JetBrains* [online]. [vid. 2023-05-19]. Dostupné z: https://www.jetbrains.com/pycharm/features/

[8] ZORIANA. Spyder - Scientific PYthon Development EnviRonment. *Quintagroup* [online]. [vid. 2023-05-19]. Dostupné z: https://quintagroup.com/cms/python/spyder

[9] PYTHON, Real. *Jupyter Notebook: An Introduction – Real Python* [online]. [vid. 2023-05-22]. Dostupné z: https://realpython.com/jupyter-notebook-introduction/

[10] TEAM, DataFlair. Advantages and Disadvantages of Python - How it is dominating Programming World. *DataFlair* [online]. 2. leden 2018 [vid. 2023-05-19]. Dostupné z: https://data-flair.training/blogs/advantages-and-disadvantages-of-python/

[11] Built-in Types. *Python documentation* [online]. [vid. 2023-05-19]. Dostupné z: https://docs.python.org/3/library/stdtypes.html

[12] 15. Floating Point Arithmetic: Issues and Limitations. *Python documentation* [online]. [vid. 2023-05-19]. Dostupné z: https://docs.python.org/3/tutorial/floatingpoint.html

[13] 3. An Informal Introduction to Python. *Python documentation* [online]. [vid. 2023-05-19]. Dostupné z: https://docs.python.org/3/tutorial/introduction.html

[14] 5. Data Structures. *Python documentation* [online]. [vid. 2023-05-19]. Dostupné z: https://docs.python.org/3/tutorial/datastructures.html

[15] PYTHON, Real. *Variables in Python – Real Python* [online]. [vid. 2023-05-19]. Dostupné z: https://realpython.com/python-variables/

[16] PYTHON, Real. *Operators and Expressions in Python – Real Python* [online]. [vid. 2023-05-19]. Dostupné z: https://realpython.com/python-operators-expressions/

[17] 6. Expressions. *Python documentation* [online]. [vid. 2023-05-19]. Dostupné z: https://docs.python.org/3/reference/expressions.html

[18] 4. More Control Flow Tools. *Python documentation* [online]. [vid. 2023-05-19]. Dostupné z: https://docs.python.org/3/tutorial/controlflow.html

[19] 8. Compound statements. *Python documentation* [online]. [vid. 2023-05-19]. Dostupné z: https://docs.python.org/3/reference/compound_stmts.html

[20] 9. Classes. *Python documentation* [online]. [vid. 2023-05-19]. Dostupné z: https://docs.python.org/3/tutorial/classes.html

[21] 3. Data model. *Python documentation* [online]. [vid. 2023-05-19]. Dostupné z: https://docs.python.org/3/reference/datamodel.html

[22] abc — Abstract Base Classes. *Python documentation* [online]. [vid. 2023-05-19]. Dostupné z: https://docs.python.org/3/library/abc.html

[23] Top 15+ Python IDEs in 2023: Choosing The Best One. *Simplilearn.com* [online]. [vid. 2023-05-19]. Dostupné z: https://www.simplilearn.com/tutorials/python-tutorial/python-ide

[24] Top 10 Python Libraries You Must Know In 2023. *Edureka* [online]. 17. prosinec 2018 [vid. 2023-05-19]. Dostupné z: https://www.edureka.co/blog/python-libraries/

[25] *What is NumPy? — NumPy v1.24 Manual* [online]. [vid. 2023-05-19]. Dostupné z: https://numpy.org/doc/stable/user/whatisnumpy.html

[26] *pandas - Python Data Analysis Library* [online]. [vid. 2023-05-19]. Dostupné z: https://pandas.pydata.org/about/

[27] *Package overview — pandas 2.0.1 documentation* [online]. [vid. 2023-05-19]. Dostupné z: https://pandas.pydata.org/docs/getting_started/overview.html

[28] *Pyplot tutorial — Matplotlib 3.7.1 documentation* [online]. [vid. 2023-05-19]. Dostupné z: https://matplotlib.org/stable/tutorials/introductory/pyplot.html

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: DVD

Přiložené DVD obsahuje:

- Soubor **fulltext** – text bakalářské práce ve formátu PDF/A
- Adresář **prilohy**:
    - Adresář **prezentace:**
        - Soubor **Python – Úvod.pptx**
        - Soubor **Python – Syntaxe.pptx**
        - Soubor **Python – Datové typy.pptx**
        - Soubor **Python – Podmínky.pptx**
        - Soubor **Python – Cykly.pptx**
        - Soubor **Python – Řetězce.pptx**
        - Soubor **Python – Sekvenční typy.pptx**
        - Soubor **Python – Funkce.pptx**
        - Soubor **Python – Moduly.pptx**
        - Soubor **Python – Práce se soubory.pptx**
        - Soubor **Python – Matplotlib.pptx**
        - Soubor **Python – Numpy.pptx**
        - Soubor **Python – Pandas.pptx**
        - Soubor **Python – Jupyter.pptx**
    - Adresář **videa:**
        - Soubor **Python – Úvod.mp4**
        - Soubor **Python – Syntaxe.mp4**
        - Soubor **Python – Datové typy.mp4**
        - Soubor **Python – Podmínky.mp4**
        - Soubor **Python – Cykly.mp4**
        - Soubor **Python – Řetězce.mp4**
        - Soubor **Python – Sekvenční typy.mp4**
        - Soubor **Python – Funkce.mp4**
        - Soubor **Python – Moduly.mp4**
        - Soubor **Python – Práce se soubory.mp4**
        - Soubor **Python – Matplotlib.mp4**
        - Soubor **Python – Numpy.mp4**
        - Soubor **Python – Pandas.mp4**

        - Soubor **Python – Jupyter.mp4**
    - Adresář **vypracovane_ulohy:**
        - Soubor **datove_typy.py**
        - Soubor **podminky.py**
        - Soubor **cykly.py**
        - Soubor **retezce.py**
        - Soubor **sekvencni_typy.py**
        - Soubor **funkce.py**
        - Soubor **práce_se_soubory.py**
        - Soubor **matplotlib.py**
        - Soubor **numpy.py**
        - Soubor **pandas.py**