UNIVERZITA PALACKÉHO V OLOMOUCI

PEDAGOGICKÁ FAKULTA

Katedra technické a informační výchovy

Bakalářská práce

Matěj Bednařík

Využití AI při výuce programování na ZŠ

Olomouc 2024

vedoucí práce: doc. RNDr. Petr Šaloun, Ph.D.

Anotace

Jméno a příjmení:	Matěj Bednařík
Katedra:	Katedra technické a informační výchovy
Vedoucí práce:	doc. RNDr. Petr Šaloun, Ph.D.
Rok obhajoby:	2024

Název práce:	Využití AI při výuce programování na ZŠ
Název v angličtině:	Using AI in teaching programming at primary school
Anotace práce:	Cílem práce je analyzovat možnosti využití komunikujících nástrojů AI při výuce programování na základních školách. V teoretické části práce budou vymezeny hlavní pojmy a vzdělávací příležitosti umělé inteligence. Jedním z cílů práce je popsat metodiku nového přístupu využití AI nástrojů při výuce, a to na základě vzorových příkladů použití využitelných pro cílovou skupinu. Vzorové programy budou vytvořeny ve dvou populárních programovacích jazycích Python a Scratch na základě stejného zadání. Takto vygenerované příklady následně kriticky zhodnotím s přihlédnutím k výukovému potenciálu.
Klíčová slova:	umělá inteligence, vzdělávací technologie, programování, základní škola, python, Scratch, chatbot, generativní umělá inteligence

Anotace v angličtině:	The aim of this paper is to analyze the possibilities of using communicating AI tools in teaching programming in primary schools. In the theoretical part of the thesis the main concepts and educational opportunities of artificial intelligence will be defined. One of the aims of the thesis is to describe the methodology of a new approach of using AI tools in teaching, based on sample examples of use applicable to the target group.The sample programs will be created in two popular programming languages, Python and Scratch, based on the same assignment.I will then critically evaluate the examples generated in this way with respect to their teaching potential.
Klíčová slova v angličtině:	artificial intelligence, educational technology, programming, elementary school, python, Scratch, chatbot, generative artificial intelligence
Přílohy vázané v práci:	1 příloha: *ChatGPT 4o vygenerované kódy v Pythonu*
Rozsah práce:	53 stran
Jazyk práce:	čeština

UNIVERZITA PALACKÉHO V OLOMOUCI
Pedagogická fakulta
Akademický rok: 2022/2023

Studijní program: Informační technologie se zaměřením na vzdělávání (maior), Matematika se zaměřením na vzdělávání (minor)
Forma studia: Kombinovaná

Podklad pro zadání BAKALÁŘSKÉ práce studenta

Jméno a příjmení:	**Matěj BEDNAŘÍK**
Osobní číslo:	**D200398**
Adresa:	Bezručova 1081/18, Kyjov, 69701 Kyjov 1, Česká republika
Téma práce:	Využití AI při výuce programování na ZŠ
Téma práce anglicky:	The use of AI in teaching programming at elementary schools
Jazyk práce:	Čeština
Vedoucí práce:	doc. RNDr. Petr Šaloun, Ph.D. Katedra technické a informační výchovy

Zásady pro vypracování:

Cílem práce je analyzovat možnosti využití komunikujících nástrojů AI při výuce programování na základních školách. V teoretické části práce budou vymezeny hlavní pojmy a vzdělávací příležitosti umělé inteligence.
Jedním z cílů práce je popsat metodiku nového přístupu využití AI nástrojů při výuce, a to na základě vzorových příkladů použití využitelných pro cílovou skupinu. Vzorové programy budou vytvořeny ve dvou populárních programovacích jazycích Python a Scratch na základě stejného zadání.
Takto vygenerované příklady následně kriticky zhodnotím s přihlédnutím k výukovému potenciálu.

Seznam doporučené literatury:

Ben Stephenson: The Python Workbook, A Brief Introduction with Exercises and Solutions.
ISBN 978-3-319-14239-5
DOI 10.1007/978-3-319-14240-1
ISBN 978-3-319-14240-1
(eBook)
Library of Congress Control Number: 2014957402
Springer Cham Heidelberg New York Dordrecht London
© Springer International Publishing Switzerland 2014
John Paul Mueller and Luca Massaron: Machine Learning for dummies
Published by: John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com
Copyright © 2016 by John Wiley & Sons, Inc., Hoboken, New Jersey
ISBN: 978-1-119-24551-3
GAVORA, Peter. Úvod do pedagogického výzkumu. 2., rozš. Brno: Paido, 2010. ISBN 978–80-7315-185-0.
HENDL, Jan a Jiří REMR. Metody výzkumu a evaluace. Praha: Portál, 2017. ISBN 978-80–262-1192-1.
BRDIČKA, Bořivoj. Digitální kompetence pro wellbeing [online]. NPI: Metodický portál RVP.CZ, Praha, 2021 [cit. 2022-11-15]. Dostupné z: https://clanky.rvp.cz/clanek/c/Z/22858/digitalni-kompetence-pro-wellbeing.html?rate=1

Stav schvalování: Vedoucím katedry schválen studentův podklad VŠKP

Podpis studenta: Datum:

Podpis vedoucího práce: Datum:

Podpis vedoucího pracoviště: Datum:

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a uvedl jsem v ní veškerou literaturu a ostatní informační zdroje, které jsem použil.

V Olomouci dne 17. 06. 2024 .....

vlastnoruční podpis

Poděkování

Chtěl bych v první řadě poděkovat mé rodině za bezmeznou trpělivost. Děkuji také vedoucímu mé práce doc. RNDr. Petru Šalounovi, Ph.D., za velmi vstřícný přístup a erudici.

Matěj Bednařík

Obsah

Úvod ... 9
1 Teoretická část ... 11
1.1 Základní pojmy a principy AI ... 11
1.1.1 Definice AI ... 11
1.1.2 Historie AI v kontextu vzdělávání ... 13
1.2 Výhody, hrozby a výzvy AI ve vzdělávání ... 15
1.2.1 Výhody AI ve vzdělávání ... 15
1.2.2 Výzvy AI ve vzdělávání ... 16
1.2.3 Hrozby AI ve vzdělávání ... 18
1.3 Vzdělávací teorie, metody a nástroje ... 19
1.3.1 Přehled nástrojů AI používaných ve vzdělávání ... 20
1.3.2 Konstruktivismus a aktivní učení ... 22
1.4 Programování jako vzdělávací disciplína ... 23
1.5 Vlastnosti dobrého kódu ... 24
1.6 Doporučení pro psaní promptů při zadávání programovacích úloh ... 28
2 Praktická část ... 31
2.1 Vývoj vzorových programů ... 33
2.1.1 Pozdrav ... 33
2.1.2 Plocha pole ... 34
2.1.3 Výpočet BMI ... 34
2.1.4 Je číslo sudé nebo liché? ... 35
2.1.5 Roční období dle data ... 36
2.1.6 Je řetězec palindrom? ... 38
2.1.7 Binární číslo na desítkové ... 39
2.1.8 Je číslo prvočíslo? ... 41
2.1.9 Palindrom rekurentně ... 43
2.1.10 Dělitelé čísla pomocí pole ... 44
2.2 Analýza a porovnání programů ... 45
2.3 Zhodnocení výukového potenciálu ... 46
Závěr ... 48
Seznam použitých zdrojů ... 49
Seznam obrázků ... 52
Seznam příloh ... 53

Úvod

Cílem práce je analyzovat možnosti využití komunikujících nástrojů umělé inteligence (AI) při výuce programování na základních školách. V teoretické části práce budou vymezeny hlavní pojmy a vzdělávací příležitosti umělé inteligence. V provázanosti s výukou programování se teoretická část bude také zabývat specifiky výuky programování a zapojování AI do výuky. Pomocí odborných zdrojů se budu snažit charakterizovat možné výhody, nevýhody a výzvy využívání AI ve vzdělávání.

Jedním z cílů práce je popsat metodiku nového přístupu využití AI nástrojů při výuce, a to na základě vzorových příkladů použití využitelných pro cílovou skupinu. Vzorové programy budou vytvořeny ve dvou populárních programovacích jazycích Python a Scratch na základě stejného zadání.

Takto vygenerované příklady následně kriticky zhodnotím s přihlédnutím k výukovému potenciálu.

V posledních době se umělá inteligence stala nedílnou součástí mnoha aspektů každodenního života, včetně vzdělávání. Její integrace do edukačních procesů otevírá nové příležitosti pro učitele i studenty, ve sféře výuky informačních systémů obzvláště.

AI přináší revoluční změny ve způsobu, jakým jsou edukační obsahy předávány a zpracovávány. Podle Holmes et al. (2019), AI umožňuje personalizaci vzdělávacích zkušeností tím, že se adaptuje na individuální potřeby a tempo každého studenta. Tento přístup nejenže podporuje zapojení a motivaci žáků, ale také zvyšuje efektivitu učení tím, že nabízí materiály přizpůsobené jejich specifickým schopnostem a předchozím znalostem.

Specificky v oblasti výuky programování na základních školách má AI potenciál zásadně ovlivnit jak obsah, tak metody výuky. Programy a aplikace založené na AI mohou sloužit jako interaktivní nástroje, které učí děti základním principům programování skrze hravou formu. Platformy jako Scratch nebo Tynker, které integrují prvky AI, umožňují studentům snadněji pochopit složité koncepty tím, že je transformují do vizuálně přitažlivých a intuitivních projektů.

Přestože jsou možnosti AI vzdělávání mimořádné, je třeba se vypořádat i s určitými výzvami, jako jsou etické dilemata, ochrana soukromí a data studentů. Etické zásady využívání

AI ve školství musí být pečlivě zváženy a řízeny, aby se zajistilo, že technologie bude sloužit k prospěchu všech zúčastněných stran bez zbytečného rizika zneužití.

Výhled do budoucnosti naznačuje, že AI bude hrát stále větší roli v edukaci. Inovace v AI a její aplikace ve školních programech nabízí příslib nejen v rozšíření edukativních nástrojů, ale také v překonání tradičních bariér, jako jsou jazykové rozdíly a geografická izolace. Důležité bude, aby vzdělávací politika a pedagogické metody držely krok s rychlým rozvojem technologií tak, aby mohly co nejefektivněji využít jejich potenciál.

Umělá inteligence znamená pro oblast vzdělávání nejen významný posun v metodách a přístupech k výuce, ale představuje i klíčový nástroj pro rozvoj digitální gramotnosti mezi mladými studenty. Jak se technologie neustále vyvíjí, bude zásadní sledovat její vliv a možnosti pro vzdělávání.

1 Teoretická část

V teoretické části práce se zaměřím na vymezení základních pojmů související s umělou inteligencí. Rozebereme její příležitosti, výhody, či nevýhody. Charakterizuji i některé vhodné vzdělávací metody, teorie a nástroje, které se zařazením AI do výuky programování mohou souviset. Teoretická část nám poslouží i jako základ pro praktickou část, proto je v ní věnována pozornost i specifikům programování jako vzdělávací disciplíně a příklady dobré praxe, díky kterým můžeme lépe vyhodnotit úkoly v praktické části.

1.1 Základní pojmy a principy AI

1.1.1 Definice AI

Umělá inteligence je široký a komplexní obor, který zahrnuje různé technologie a metodiky umožňující strojům vykonávat úkoly, které by běžně vyžadovaly lidskou inteligenci. Tato kapitola se zaměřuje na různé definice AI, které nabízejí pohledy různých odborníků a institucí.

Podle Russella a Norviga (2016) lze AI definovat jako „studium agentů, kteří přijímají vnější podněty a provádějí akce, které maximalizují jejich šance na úspěch“. Tato definice zdůrazňuje schopnost AI adaptovat se na změny v prostředí a rozhodovat se na základě dostupných informací, což je klíčové pro její aplikace ve vzdělávání.

Encyclopaedia Britannica popisuje AI jako schopnost digitálního počítače nebo počítačem řízeného robota vykonávat úkoly běžně spojované s inteligentními bytostmi, jako je schopnost uvažovat, objevovat význam, generalizovat nebo se učit z minulých zkušeností. Tento pohled podtrhuje široké spektrum schopností, které AI může mít, a její potenciál nahradit nebo doplnit lidské schopnosti ve vzdělávacích procesech (Copeland, 2024).

Stanford University (2023) definuje AI jako obor zaměřený na vytváření systémů, které mohou vykonávat úkoly, jež vyžadují lidskou inteligenci. Tato definice zahrnuje různé přístupy, jako je strojové učení, hluboké učení a zpracování přirozeného jazyka, které jsou klíčové pro moderní vzdělávací aplikace. AI může například analyzovat data o výkonnosti studentů a na základě těchto dat přizpůsobit výuku, což zvyšuje efektivitu vzdělávání.

Podle Caltech Science Exchange (2023) se AI zabývá tvorbou strojů, které mohou napodobovat lidské myšlení. Tento přístup zahrnuje analýzu toho, jak mozek zpracovává informace, a vývoj technologií, které mohou vykonávat nové úkoly a přizpůsobovat se novým situacím. AI aplikace využívají data k identifikaci vzorců, provádění predikcí a rozhodování, často s větší efektivitou a přesností než lidé.

One Hundred Year Study on Artificial Intelligence (AI100) poskytuje užitečné rozdělení AI na různé typy, od reaktivních strojů až po stroje s vlastním vědomím. Toto rozdělení pomáhá pochopit různé úrovně schopností, které AI může mít, a jak mohou být tyto schopnosti aplikovány ve vzdělávání. Například reaktivní stroje mohou být použity pro jednoduché úkoly, zatímco systémy s teorií mysli mohou interagovat s uživateli a rozumět jejich potřebám a emocím (AI100, 2023).

Strojové učení, jako jedna z klíčových technologií AI, umožňuje systémům učit se a zlepšovat na základě zkušeností. Algoritmy strojového učení analyzují velké množství dat, aby identifikovaly vzorce a vytvářely modely pro predikci a rozhodování. Tato technologie je zásadní pro vývoj adaptivních vzdělávacích systémů, které mohou personalizovat výukový proces podle individuálních potřeb studentů (Jordan & Mitchell, 2015).

Hluboké učení, podmnožina strojového učení, využívá umělé neuronové sítě inspirované lidským mozkem k rozpoznávání složitých vzorců a provádění intenzivních úkolů zpracování dat. Vzdělávací aplikace hlubokého učení mohou například analyzovat studentské odpovědi a poskytovat okamžitou a přizpůsobenou zpětnou vazbu, což výrazně zlepšuje učební výsledky.

Generativní nástroje AI, mají schopnost vytvářet nový obsah, včetně textů, obrázků a hudby. Tyto technologie mohou být využity ve vzdělávání k vytváření interaktivních a přizpůsobitelných materiálů, které simulují reálné situace a podporují kritické myšlení a kreativitu studentů (Goodfellow et al., 2014).

Jedná se jednoduše o velmi komplexní pojem, na který existuje mnoho náhledů. V rámci této práce ale budeme vnímat AI především jako generativní nástroj reprezentovaný například generátory textu, hlasu, obrazu a podobně.

1.1.2 Historie AI v kontextu vzdělávání

Umělá inteligence (AI) prošla od svého vzniku v polovině 20. století dramatickým vývojem, který významně ovlivnil mnoho oblastí, včetně vzdělávání. AI se od počátečních experimentů ve výzkumných laboratořích postupně dostala do škol a vzdělávacích institucí, kde přináší inovativní řešení a nové možnosti pro výuku a učení.

První koncepty AI se objevily v roce 1956 na konferenci v Dartmouth College, kde vědci jako John McCarthy, Marvin Minsky, Nathaniel Rochester a Claude Shannon diskutovali o možnostech vytvoření strojů schopných vykazovat inteligentní chování. Během následujících desetiletí se AI rozvíjela především v akademickém prostředí, kde byly kladeny základy pro vývoj algoritmů a modelů, které dnes používáme (Russell & Norvig, 2016).

První aplikace AI ve vzdělávání se objevily již v 60. letech 20. století, kdy byly vyvinuty první inteligentní výukové systémy (ITS). Tyto systémy byly navrženy tak, aby poskytovaly personalizovanou zpětnou vazbu a podporu studentům při jejich učení. Například systém SCHOLAR, vyvinutý v 70. letech, byl jedním z prvních ITS, který používal AI k interakci se studenty a poskytování individuálních instrukcí (Carbonell, 1970).

V 80. letech došlo k významnému pokroku ve vývoji AI díky zvýšenému výpočetnímu výkonu a novým algoritmům. V oblasti vzdělávání se objevily systémy jako PLATO a GUIDON, které využívaly AI k poskytování sofistikovanějších a personalizovanějších výukových zážitků. Tyto systémy byly schopny analyzovat studentovy odpovědi a přizpůsobovat instrukce na základě jeho výkonu (Wenger, 1987). V 90. letech se AI začala více prosazovat v komerční sféře, což vedlo k vývoji nových nástrojů a aplikací pro vzdělávání. Jedním z významných příkladů je systém Cognitive Tutor, vyvinutý na Carnegie Mellon University, který poskytoval adaptivní výuku matematiky založenou na analýze studentova chování a výkonu (Anderson et al., 1995).

Od začátku 21. století se AI ve vzdělávání dále rozvíjí a stává se stále sofistikovanější. S nástupem technologií jako je strojové učení a hluboké učení se výrazně zlepšila schopnost AI systémů analyzovat data a poskytovat personalizovanou podporu studentům. Moderní AI

aplikace ve vzdělávání zahrnují inteligentní tutoringové systémy, adaptivní výukové platformy a nástroje pro automatizované hodnocení.

Inteligentní tutoringové systémy, jako je ALEKS, využívají AI k poskytování individuální podpory studentům v reálném čase. Tyto systémy sledují pokrok studentů, analyzují jejich chyby a navrhují přizpůsobené výukové aktivity, které pomáhají zlepšit jejich dovednosti a znalosti (Fletcher & Atkinson, 2015). Adaptivní výukové platformy, jako je Knewton, používají AI k personalizaci vzdělávacích materiálů na základě analýzy dat o studentově výkonnosti a preferencích. Tyto platformy mohou dynamicky přizpůsobovat obsah a tempo výuky tak, aby co nejlépe vyhovovaly individuálním potřebám každého studenta.

Automatizované hodnocení, které využívá AI k analýze a hodnocení studentských prací, se stává stále běžnější. Tento přístup umožňuje učitelům efektivněji spravovat velké množství úkolů a poskytovat rychlou zpětnou vazbu. Například nástroj Gradescope používá AI k automatizovanému hodnocení a poskytování detailní zpětné vazby na studentské úlohy.

Budoucnost AI ve vzdělávání slibuje další inovace a rozvoj. Očekává se, že AI bude hrát klíčovou roli v personalizaci vzdělávacího procesu, zlepšení dostupnosti a kvality vzdělávacích materiálů a podpoře celoživotního učení. Kromě toho se budou AI technologie stále více integrovat do vzdělávacích systémů, což umožní učitelům a studentům plně využívat jejich potenciál. Jedním z hlavních cílů budoucího výzkumu bude také řešení etických a sociálních otázek spojených s využíváním AI ve vzdělávání. To zahrnuje zajištění spravedlivého přístupu ke vzdělávacím technologiím, ochranu soukromí studentů a prevenci předsudků v AI algoritmech (Holmes et al., 2019).

Historie a vývoj AI v kontextu vzdělávání ukazuje, jak se tato technologie postupně stává nedílnou součástí moderních vzdělávacích systémů. Od raných inteligentních výukových systémů až po moderní adaptivní platformy a automatizované hodnocení, AI přináší nové možnosti a výzvy, které mohou zásadně ovlivnit způsob, jakým se studenti učí a učitelé učí.

1.2 Výhody, hrozby a výzvy AI ve vzdělávání

1.2.1 Výhody AI ve vzdělávání

Umělá inteligence má potenciál zásadně transformovat vzdělávací procesy a přinést mnoho výhod jak pro studenty, tak pro učitele. V tomto oddíle se budu snažit charakterizovat několik z klíčových výhod, které AI může nabídnout ve vzdělávacím prostředí.

Jednou z největších výhod AI je personalizace výuky. AI umožňuje přizpůsobit učební materiály a metody individuálním potřebám a schopnostem studentů. Adaptivní výukové systémy analyzují výkonnost studentů a na základě této analýzy upravují obsah a tempo výuky. Tento přístup zvyšuje efektivitu učení, protože každý student dostává přesně to, co potřebuje k dosažení svých vzdělávacích cílů (Holmes et al., 2019). Personalizované učení nejenže zvyšuje angažovanost studentů, ale také zlepšuje jejich výkonnost a motivaci.

Další výhodou může být automatizace hodnocení. AI systémy mohou automaticky hodnotit studentské práce a domácí úkoly, což šetří čas učitelům a umožňuje jim více se soustředit na individuální pomoc studentům. Automatizované hodnocení může být také přesnější a konzistentnější než lidské hodnocení, čímž se minimalizuje subjektivita a chyby (Luxton-Reilly, 2016). AI umožňuje poskytovat okamžitou zpětnou vazbu, což studentům pomáhá rychleji se učit z vlastních chyb a zlepšovat své dovednosti.

AI může podporovat rozvoj kritického myšlení a kreativity. Interaktivní výukové nástroje poskytují studentům příležitosti k řešení komplexních problémů a podporují jejich schopnost přemýšlet kriticky a inovativně. Simulace a modelování umožňují studentům experimentovat s různými scénáři a objevovat nové koncepty a nápady, což přispívá k jejich hlubšímu porozumění a tvůrčímu myšlení (Luckin et al., 2016).

Inkluzivita je dalším klíčovým přínosem AI ve vzdělávání. AI může přizpůsobit výukové materiály a metody pro studenty se specifickými vzdělávacími potřebami, čímž zajišťuje rovný přístup k výuce. Kupříkladu může AI poskytovat vizuální a sluchové pomůcky

pro studenty s postižením nebo překládat výukové materiály do různých jazyků pro studenty, kteří nejsou rodilými mluvčími. Tímto způsobem AI napomáhá k vytváření inkluzivního vzdělávacího prostředí, ve kterém má každý student šanci uspět.

Další potenciální výhodou AI je efektivita a úspora času. AI nástroje mohou automatizovat rutinní administrativní úkoly, jako je plánování rozvrhů, sledování docházky nebo správa studijních materiálů. To umožňuje učitelům věnovat více času výuce a interakci se studenty. Kromě toho AI může poskytovat učitelům cenné analytické nástroje a data, které jim pomohou lépe porozumět potřebám studentů a efektivněji plánovat výukové aktivity.

V neposlední řadě AI podporuje vzdělávání na dálku a hybridní výuku. V době, kdy je online vzdělávání stále důležitější, AI nástroje umožňují vytvářet interaktivní a motivující vzdělávací zkušenosti i mimo tradiční třídy. Online platformy využívající AI mohou poskytovat personalizované lekce, sledovat pokrok studentů a poskytovat zpětnou vazbu v reálném čase, což zajišťuje kontinuální a efektivní učení bez ohledu na fyzickou přítomnost.

Integrace AI do vzdělávání nabízí mnoho výhod, které mohou výrazně zlepšit kvalitu a efektivitu vzdělávacího procesu. Od personalizace výuky a automatizace hodnocení po podporu kritického myšlení a inkluzivity, AI poskytuje nástroje a zdroje, které mohou podpořit studenty v dosažení jejich vzdělávacích cílů a učitelům usnadnit jejich práci.

1.2.2 Výzvy AI ve vzdělávání

Integrace umělé inteligence do vzdělávacích procesů přináší také řadu výzev, které je třeba pečlivě zvážit a řešit. Tyto výzvy se týkají jak pedagogických aspektů, tak technických a etických otázek. V tomto oddíle postihneme některé výzvy spojené s využíváním AI ve výuce.

Jedna z hlavních výzev bude vzdělání učitelů. Pro efektivní využívání AI ve výuce je nezbytné, aby učitelé měli dostatečné znalosti a dovednosti v této oblasti. To zahrnuje nejen technické aspekty, jako je porozumění fungování AI systémů, ale také schopnost integrovat tyto technologie do svých výukových metod a plánů. Učitelé potřebují kontinuální profesní rozvoj a školení, aby mohli plně využít potenciál AI a zároveň minimalizovali rizika spojená s jejím používáním (Holmes et al., 2019).

Zavádění AI do vzdělávání vyžaduje nové metody a restrukturalizaci kurikula. Tradiční výukové metody nemusí být vždy kompatibilní s AI technologiemi, což vyžaduje inovativní přístupy k výuce a hodnocení. Kurikulum musí být přizpůsobeno tak, aby zahrnovalo nejen technické dovednosti potřebné pro práci s AI, ale také kritické myšlení, kreativitu a etické otázky spojené s používáním AI. Tento proces může být časově náročný a vyžaduje spolupráci mezi pedagogy, technickými experty a výzkumníky.

Takzvaná Prompt proficiency, tedy schopnost efektivně komunikovat s AI systémy prostřednictvím zadávání správných příkazů a dotazů, bude další z nových klíčových schopností. Studenti a učitelé musí být schopni formulovat přesné a relevantní otázky, aby mohli plně využít schopností AI. Tato dovednost je nezbytná pro efektivní využití AI ve výuce, proto se jí budeme věnovat v samostatném oddíle.

Integrace AI do vzdělávání přináší také etické otázky. Otázky soukromí a bezpečnosti dat, spravedlnosti a odpovědnosti. AI systémy často shromažďují a analyzují velké množství osobních údajů o studentech, což vyvolává obavy ohledně ochrany soukromí. Kromě toho mohou být AI systémy zaujaté, což může vést k nespravedlivému zacházení se studenty. Bude nejspíš nezbytné zavést robustní etické a právní rámce, které zajistí ochranu práv studentů a minimalizují rizika spojená s používáním AI (Calo, 2017).

Zatímco AI může výrazně zlepšit efektivitu a personalizaci výuky, je důležité, aby podporovala kritické myšlení studentů. Existuje riziko, že studenti budou příliš závislí na AI a nebudou rozvíjet své vlastní analytické a rozhodovací schopnosti. Podle Selwyna (2019) musí učitelé zajistit, aby AI nástroje byly používány jako doplněk k tradičním výukovým metodám, nikoli jako jejich náhrada.

Další významnou výzvou je zajištění rovného přístupu k AI technologiím ve vzdělávání. Existuje riziko, že nerovnosti v přístupu k technologiím mohou prohloubit stávající vzdělávací nerovnosti. Studenti z méně privilegovaných prostředí mohou mít omezený přístup k moderním technologiím a digitálním zdrojům, což může negativně ovlivnit jejich vzdělávací výsledky. Je nezbytné zajistit, aby všechny školy a studenti měli přístup k potřebným technologiím a aby byly implementovány programy na podporu digitální gramotnosti.

Integrace AI do vzdělávání nabízí mnoho příležitostí, ale také přináší řadu výzev, které je třeba pečlivě řešit. Od vzdělávání učitelů a restrukturalizace kurikula po etické otázky a zajištění rovného přístupu, je nezbytné, aby vývoj a implementace AI technologií ve vzdělávání byly provázeny robustními právními a etickými rámce. Tím se zajistí, že AI bude sloužit jako nástroj pro zlepšení kvality vzdělávání a podpoří rovný přístup ke vzdělávacím příležitostem pro všechny studenty.

1.2.3 Hrozby AI ve vzdělávání

Integrace umělé inteligence do vzdělávání nabízí řadu výhod, ale také významné hrozby, které je třeba pečlivě zvážit. Tyto hrozby mohou ovlivnit kvalitu vzdělávání a zasahovat do práv a bezpečnosti studentů a učitelů.

Jedním z hlavních rizik je takzvaný bias (předsudky) v AI systémech. AI systémy se učí z historických dat, která mohou obsahovat předsudky. Pokud jsou tato data zaujatá, mohou AI systémy tyto předsudky reprodukovat a zesilovat. To může vést k nespravedlivému zacházení se studenty na základě rasy, pohlaví, socioekonomického statusu nebo jiných charakteristik (O'Neil, 2016). Například automatizované hodnocení může nesprávně hodnotit práce studentů, pokud je trénováno na datech, která upřednostňují určitý styl nebo jazyk. Obecně mohou AI systémy upřednostňovat informace, které nejsou reflektivní vůči rozmanitosti studentů.

Významným rizikem je také ochrana soukromí. AI systémy často shromažďují a analyzují velké množství osobních údajů o studentech, včetně jejich výkonnosti, chování a zdravotních záznamů. Podle Westa (2019) mohou tato data být zranitelná vůči zneužití, ať už jde o neoprávněný přístup, hackování nebo prodej třetím stranám. Porušení soukromí může mít závažné důsledky pro studenty a jejich rodiny, a neoprávněné sledování může vést k pocitu nedůvěry a omezení osobní svobody.

Další problém spočívá v nejasných právních rámcích. Současné právní rámce často nejsou dostatečně připraveny na rychlý rozvoj AI technologií. To může vést k nejasnostem ohledně odpovědnosti za chyby, ochrany duševního vlastnictví a práva na soukromí (Calo, 2017). Například otázka právní odpovědnosti, kdy není jasné, kdo nese odpovědnost za chyby AI v rozhodování nebo hodnocení, a problematika duševního vlastnictví, kde je třeba řešit otázky autorských práv u materiálů vytvořených AI.

AI může také představovat ohrožení pracovního trhu. Automatizace mnoha úkolů, které dnes vykonávají učitelé a další vzdělávací pracovníci, může vést k obavám z nahrazení lidských pracovníků stroji a ztrátě pracovních míst (Frey & Osborne, 2017). Automatizace administrativních a některých výukových úkolů může vést ke snížení počtu pracovních míst pro učitele a administrátory, a učitelé mohou být nuceni přizpůsobit se novým technologiím a převzít nové role, což vyžaduje nové dovednosti a školení.

Nadměrné spoléhání se na AI může narušit tradiční pedagogické vztahy mezi učiteli a studenty, což představuje další hrozbu. Osobní interakce a mentorský vztah mohou být oslabené, což může negativně ovlivnit kvalitu vzdělávání (Selwyn, 2019). Studenti mohou postrádat osobní podporu a vedení od učitelů, což je zásadní pro jejich emocionální a sociální vývoj, a automatizace může vést k vnímání vzdělávacího procesu jako mechanického a neosobního.

Integrace AI do vzdělávání přináší mnoho příležitostí, ale také významné hrozby, které je třeba pečlivě řídit a regulovat. Vývoj a implementace AI technologií ve vzdělávání musí být provázeny robustními právními a etickými rámci, které zajistí ochranu práv studentů a učitelů a minimalizují rizika spojená s jejich používáním.

1.3 Vzdělávací teorie, metody a nástroje

Integrace umělé inteligence do výuky nejen programování může výrazně obohatit učební proces a nabídnout nové možnosti personalizace a interaktivity. V této podkapitole se budu snažit postihnout, které vzdělávací metody, teorie a nástroje mohou být obzvláště přínosné při výuce programování.

Adaptivní učení je metoda, která využívá technologie k přizpůsobení obsahu a tempa výuky individuálním potřebám studentů. AI může analyzovat výkonnost studentů a na základě toho upravovat úroveň obtížnosti úloh, poskytovat doplňkové materiály nebo měnit přístup k výuce. Adaptivní výukové systémy mohou studentům nabídnout úkoly odpovídající jejich aktuálním dovednostem a znalostem, čímž maximalizují jejich učební potenciál (Holmes et al., 2019).

Projektově orientované učení umožňuje studentům získat praktické zkušenosti s programováním prostřednictvím řešení reálných problémů a tvorby projektů. AI může podporovat tento proces poskytováním nástrojů a zdrojů, které studentům pomáhají plánovat, implementovat a vyhodnocovat jejich projekty. Třeba online platforma pro výuku programování Scratch, které se budeme věnovat v praktické části, umožňuje studentům vytvářet interaktivní příběhy, hry a animace, čímž podporuje jejich kreativitu a praktické dovednosti.

Dalším důležitým pojmem je gamifikace. Ta přináší herní prvky do vzdělávání s cílem zvýšit motivaci a zapojení studentů. Díky umělé inteligenci lze vytvářet personalizované herní zážitky, které se přizpůsobují individuálním schopnostem a pokroku studentů. Hry jako CodeCombat učí programování prostřednictvím interaktivních herních úkolů, které studenty motivují a udržují jejich zájem o učení.

Blended learning kombinuje tradiční výuku v učebně s online vzdělávacími aktivitami. AI může zajišťovat synchronizaci obou forem výuky, poskytovat interaktivní materiály a sledovat pokrok studentů. Online platformy jako Coursera a Udacity nabízejí kurzy programování, které mohou být integrovány do tradičních výukových plánů, čímž kombinují výhody obou přístupů.

Integrace AI do výuky nabízí mnoho výhod a metod, které mohou zlepšit efektivitu a kvalitu vzdělávacího procesu. Od adaptivního učení a inteligentních tutoringových systémů po projektově orientované učení a gamifikaci, AI poskytuje nástroje a zdroje, které mohou podpořit studenty v rozvoji jejich programovacích dovedností.

1.3.1 Přehled nástrojů AI používaných ve vzdělávání

Umělá inteligence se stále více integruje do vzdělávacích systémů, kde nabízí nové možnosti pro personalizaci výuky, zlepšení efektivity učení a poskytování podpory studentům a učitelům. AI nástroje používané ve vzdělávání lze rozdělit do několika typů podle jejich specifických funkcí a aplikací.

Inteligentní tutoringové systémy jsou navrženy tak, aby poskytovaly individuální podporu studentům v reálném čase. Tyto systémy sledují pokrok studentů, analyzují jejich

chyby a nabízejí přizpůsobené výukové aktivity. Příkladem ITS je systém ALEKS, který využívá AI k diagnostice znalostí studentů a poskytování personalizovaných výukových cest. ITS mohou výrazně zlepšit efektivitu učení tím, že poskytují okamžitou zpětnou vazbu a přizpůsobují se individuálním potřebám studentů (Fletcher & Atkinson, 2015).

Automatizované hodnocení využívá AI k analýze a hodnocení studentských prací. Tento přístup umožňuje učitelům efektivněji spravovat velké množství úkolů a poskytovat rychlou zpětnou vazbu. Gradescope je nástroj, který používá AI k automatizovanému hodnocení a poskytování detailní zpětné vazby na studentské úlohy. AI systémy pro automatizované hodnocení mohou analyzovat textové odpovědi, programovací úlohy a dokonce i esejové práce (Singh et al., 2019).

Chatboti a virtuální asistenti poskytují podporu studentům a učitelům prostřednictvím interaktivních rozhovorů. Tyto nástroje mohou odpovídat na otázky, poskytovat informace o kurzech a materiálech, a dokonce i pomáhat s administrativními úkoly. V posledních letech jsme svědky výrazného nástupu AI zejména díky pokročilým jazykovým modelům, jako je ChatGPT vyvinutý společností OpenAI. Jsou to právě chatboti, kteří stojí za ohromnou popularizací AI v posledních letech. Nabízí totiž bezprostřední interakci s uživatelem v rámci intuitivní aplikace.

Systémy pro analýzu učení shromažďují a analyzují data o výkonnosti studentů za účelem zlepšení výukových procesů. Tyto systémy mohou identifikovat vzorce v chování studentů, předpovídat jejich úspěšnost a navrhovat intervence pro zlepšení výsledků. Learning analytics systémy, jako je Blackboard Analytics, využívají AI k analýze dat z různých zdrojů a poskytují učitelům a vedení školy cenné informace o pokroku a potřebách studentů (Siemens & Long, 2011).

Simulace nebo systémy virtuální reality (VR) využívají AI k vytváření realistických a interaktivních vzdělávacích prostředí. Tyto technologie umožňují studentům procvičovat dovednosti v bezpečném a kontrolovaném prostředí. VR nástroje, jako je Meta Quest 2 nebo HTC Vive, poskytují studentům možnost interagovat s virtuálními objekty a scénáři, což může zlepšit jejich pochopení složitých konceptů a dovedností.

1.3.2 Konstruktivismus a aktivní učení

Ve vztahu k novým vzdělávacím příležitostem AI stojí za krátký rozbor i pedagogický směr konstruktivismus a metody aktivního učení.

Konstruktivismus je vzdělávací teorie, která tvrdí, že učení je aktivní, konstruktivní proces. Studenti se učí nejlépe tím, že konstruují vlastní porozumění a znalosti prostřednictvím zkušeností a reflexe. Tato teorie byla silně ovlivněna prací psychologů jako Jean Piaget a Lev Vygotsky. Jean Piaget zdůrazňoval, že děti konstruují své vlastní poznání prostřednictvím interakce s prostředím a že učení je proces aktivního budování a přetváření stávajících mentálních struktur (Piaget, 1972). Lev Vygotsky zase kladl důraz na sociální aspekty učení, argumentoval, že kognitivní vývoj je silně ovlivněn interakcí s ostatními a že jazyk a komunikace hrají klíčovou roli v procesu učení (Vygotsky, 1978).

Konstruktivistické principy jsou efektivní při výuce programování. Z povahy věci se žáci při tvorbě programů, nebo třeba programování robotů vzdělávají konstruktivisticky. Studenti mohou používat AI nástroje k vytváření vlastních programů a aplikací, což jim umožňuje aktivně se zapojit do procesu učení a konstruovat své vlastní porozumění programovacím konceptům. AI nástroje mohou podporovat konstruktivistické přístupy tím, že poskytují interaktivní a adaptivní výukové prostředí. Například adaptivní výukové platformy jako Knewton a DreamBox analyzují data o výkonu studentů a přizpůsobují výukové aktivity tak, aby odpovídaly jejich individuálním potřebám a úrovním dovedností. To umožňuje studentům postupovat vlastním tempem a konstruovat své vlastní porozumění programovacím konceptům.

Metody aktivního učení jsou pedagogické přístupy, které kladou důraz na aktivní účast studentů v procesu učení. Aktivní učení zahrnuje řadu technik, které podporují studenty v aktivním zkoumání, diskusi, řešení problémů a aplikaci znalostí. Tyto metody mohou zahrnovat skupinovou práci, praktické projekty, simulace a interaktivní výukové aktivity. Projektově orientované učení je metoda aktivního učení, která umožňuje studentům získat praktické zkušenosti s prostřednictvím řešení reálných problémů a tvorby projektů. AI může podporovat tento proces poskytováním nástrojů a zdrojů, které studentům pomáhají plánovat, implementovat a vyhodnocovat jejich projekty (Blumenfeld et al., 1991). Kolaborativní učení je další metoda aktivního učení, která zahrnuje spolupráci mezi studenty na řešení problémů a

dosažení společných cílů. AI může zprostředkovat tento proces tím, že identifikuje studenty s podobnými dovednostmi nebo potřebami a navrhuje efektivní způsoby spolupráce (Dillenbourg, 1999).

1.4 Programování jako vzdělávací disciplína

Programování se v posledních letech stalo klíčovou vzdělávací disciplínou, která rozvíjí nejen technické dovednosti, ale i kritické a analytické myšlení. V českém kontextu hraje významnou roli nový kurikulární dokument, známý jako „Nová informatika“, který zdůrazňuje rozvoj informatického myšlení a programování na základních a středních školách.

Informatické myšlení je koncept, který se zaměřuje na řešení problémů a návrh systémů způsobem, který je inspirován způsobem myšlení informatika. Tento přístup zahrnuje rozklad problémů na menší části, rozpoznávání vzorců, abstrakci a algoritmizaci. Je to základní dovednost, která studentům umožňuje nejen chápat a vytvářet technologie, ale také řešit komplexní problémy systematicky a logicky (Wing, 2006).

V roce 2021 byl v České republice zaveden nový kurikulární dokument nazvaný „Nová informatika“, který má za cíl integrovat informatické myšlení do vzdělávacího procesu. Tento dokument přináší významné změny ve způsobu, jakým je informatika vyučována na základních a středních školách, a zdůrazňuje význam programování jako klíčové dovednosti pro 21. století (MŠMT, 2021).

Hlavním cílem Nové informatiky je připravit studenty na digitální svět tím, že jim poskytne potřebné dovednosti a znalosti k využívání a vytváření technologií. Kurikulum je strukturováno tak, aby podporovalo rozvoj informatického myšlení, programování a tvorbu aplikací, digitální gramotnost a bezpečnost. Studenti se učí rozpoznávat a řešit problémy pomocí algoritmů a logických postupů, vytvářet jednoduché aplikace a řešit reálné problémy prostřednictvím kódování a rozumět digitální bezpečnosti a etice.

V rámci Nové informatiky se programování vyučuje od prvního stupně základní školy až po střední školu. Na prvním stupni se studenti seznamují se základními koncepty programování prostřednictvím vizuálních programovacích nástrojů, jako je Scratch. Na druhém

stupni a středních školách pak přecházejí k textovým programovacím jazykům, jako je Python, který umožňuje hlubší pochopení algoritmických principů a praktickou aplikaci programování.

Programování jako vzdělávací disciplína nabízí několik klíčových výhod. Rozvíjí kritické myšlení a schopnost řešit problémy, protože vyžaduje, aby studenti analyzovali problémy, rozkládali je na menší části a vytvářeli efektivní řešení. Tato dovednost je přenositelná i do jiných oblastí života a vzdělávání. Dále podporuje kreativitu a inovace, protože studentům umožňuje vytvářet vlastní projekty, hry a aplikace, čímž rozvíjejí své inovativní myšlení a dovednosti v oblasti designu. Navíc připravuje studenty na budoucí kariéru, protože dovednosti získané při studiu programování jsou vysoce ceněny na trhu práce a otevírají široké spektrum kariérních příležitostí v technologických a inženýrských oborech.

1.5 Vlastnosti dobrého kódu

Vhledem k tomu, že v praktické části práce se budeme snažit hodnotit výstupy kódu generovaného AI, je potřeba věnovat kapitolu i definování toho, co to vlastně dobrý kód je. Dobrý kód by měl být čitelný, udržovatelný, efektivní, robustní a dokumentovaný. Následující kapitola podrobně popisuje jednotlivé vlastnosti, které by měl dobrý kód mít, a ilustruje je konkrétními příklady.

Čitelnost je jednou z nejdůležitějších vlastností dobrého kódu. Čitelný kód je snadno pochopitelný pro jiné programátory, což usnadňuje jeho údržbu a rozvoj. Čitelnost lze dosáhnout použitím jasných a smysluplných názvů proměnných a funkcí, konzistentním formátováním a strukturou kódu a přidáním komentářů tam, kde je to nutné. Příklady kódů budou v programovacím jazyce Python.

Příklad špatně čitelného kódu:

```
def f(x):
    r = 0
    for i in range(len(x)):
        r += x[i]
    return r / len(x)
```

Příklad dobře čitelného kódu:

```
def calculate_average(numbers):
    total_sum = 0
    for number in numbers:
        total_sum += number
    return total_sum / len(numbers)
```

Druhý příklad používá smysluplné názvy funkcí a proměnných, což zvyšuje čitelnost a srozumitelnost kódu (McConnell, 2004).

Udržovatelnost znamená, že kód lze snadno upravovat a rozšiřovat bez zavádění chyb. To zahrnuje modulární strukturu kódu, kde jsou jednotlivé části kódu oddělené a nezávislé, což usnadňuje jejich úpravy a testování. Dále je důležité, aby kód byl dobře dokumentovaný a aby byly dodržovány osvědčené postupy, jako je DRY princip (Don't Repeat Yourself), který minimalizuje duplicitu kódu (Hunt & Thomas, 1999).

Příklad udržovatelného kódu:

```
def calculate_total(numbers):
    return sum(numbers)

def calculate_average(numbers):
    total = calculate_total(numbers)
    return total / len(numbers)
```

V tomto příkladu je funkce calculate_total znovu použita ve funkci calculate_average, což zvyšuje modularitu a udržovatelnost kódu.

Efektivita kódu znamená, že kód vykonává své úkoly co nejrychleji a s co nejmenšími zdroji. To zahrnuje optimalizaci algoritmů a datových struktur, minimalizaci zbytečných operací a zajištění, že kód nezpůsobuje zbytečné zatížení systému.

Příklad neefektivního kódu:

```
def find_max(numbers):
    max_number = numbers[0]
    for number in numbers:
        if number > max_number:
            max_number = number
    return max_number
```

Příklad efektivního kódu:

```
def find_max(numbers):
    return max(numbers)
```

Druhý příklad využívá vestavěnou funkci max, která je optimalizována pro výkon, což zvyšuje efektivitu kódu.

Robustní kód je odolný vůči chybám a neočekávaným vstupům. To zahrnuje validaci vstupů, zpracování výjimek a zajištění, že kód bude správně fungovat i za neobvyklých podmínek (McConnell, 2004).

Příklad robustního kódu:

```
def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        return "Chyba: dělení nulou"
```

V tomto příkladu funkce divide zpracovává výjimku ZeroDivisionError, což zajišťuje, že kód je robustní a dokáže správně reagovat na chybu dělení nulou.

Dobře dokumentovaný kód obsahuje komentáře a dokumentační řetězce, které vysvětlují účel a fungování kódu. Dokumentace by měla být jasná, stručná a relevantní, aby byla užitečná pro jiné programátory, kteří budou kód číst a udržovat (Martin, 2008).

Příklad dobře dokumentovaného kódu:

```
def calculate_average(numbers):
    """
    Tato funkce vypočítá průměrnou hodnotu seznamu čísel.

    Args:
    numbers (list): Seznam čísel

    Returns:
    float: Průměrná hodnota seznamu čísel
    """
    total_sum = sum(numbers)
    return total_sum / len(numbers)
```

V tomto příkladu je funkce calculate_average doplněna o dokumentační řetězec, který vysvětluje účel funkce, její argumenty a návratovou hodnotu.

1.6 Doporučení pro psaní promptů při zadávání programovacích úloh

Efektivní psaní promptů pro AI nástroje, jako jsou ChatGPT a Codebreaker, je důležité pro dosažení kvalitních a užitečných odpovědí. Existuje několik osvědčených doporučení, která mohou pomoci vytvořit prompty tak, aby přinášely co nejlepší výsledky.

Především je důležité, aby prompty byly jasné a konkrétní. Při zadávání úloh je nutné jasně definovat, co přesně požadujete, aby AI vykonala. Namísto obecného zadání jako "napiš program" je lepší specifikovat, co má program dělat.

Příklad:

"Napiš Python skript, který vypočítá průměrný věk ze seznamu věků. Seznam věků je uložen v proměnné vek_seznam ve formátu [20, 30, 25, 22]."

Toto zadání poskytuje dostatek detailů, aby AI mohla správně pochopit požadovaný úkol. Použití jednoduchého a srozumitelného jazyka je další klíčový prvek. Vyhněte se složitým nebo nejednoznačným formulacím, které by mohly vést k nesprávnému porozumění úkolu.

Příklad:

"Napiš funkci v Pythonu s názvem soucet, která přijme dva argumenty a vrátí jejich součet."

Poskytování kontextu je také důležité pro lepší porozumění úkolu. Pokud je to relevantní, zahrňte informace o tom, jaký problém se snažíte vyřešit nebo jaké jsou předpoklady.

Příklad:

"Představ si, že píšeš program pro školní databázi. Napiš Python skript, který vypočítá průměrné hodnocení studenta. Hodnocení jsou uložena v seznamu hodnoceni ve formátu [85, 90, 78, 92]."

Dalším doporučením je uvádět příklady vstupu a výstupu. Příklady pomáhají AI lépe porozumět, jak má správný výsledek vypadat.

Příklad:

"Napiš Python funkci max_hodnota, která přijme seznam čísel a vrátí nejvyšší hodnotu. Například pro vstup [3, 5, 2, 8, 1] by funkce měla vrátit 8."

Zohlednění potenciálních chyb a omezení je dalším tipem pro získání dobrých výsledků. Pokud očekáváte specifické chyby nebo omezení, zmiňte je v promptu. To může AI pomoci lépe přizpůsobit odpověď vašim potřebám.

Příklad:

"Napiš Python funkci deleni, která přijme dva argumenty a vrátí výsledek jejich dělení. Ošetři případ dělení nulou a vrať v takovém případě text 'Chyba: dělení nulou'."

Iterativní přístup je rovněž užitečný. Někdy je nutné upravit prompt na základě předchozích odpovědí. Nebojte se iterovat a zdokonalovat své prompty podle toho, jak AI reaguje. Každá iterace vám může poskytnout lepší pochopení toho, jak AI funguje a jaké informace potřebuje.

Příklad:

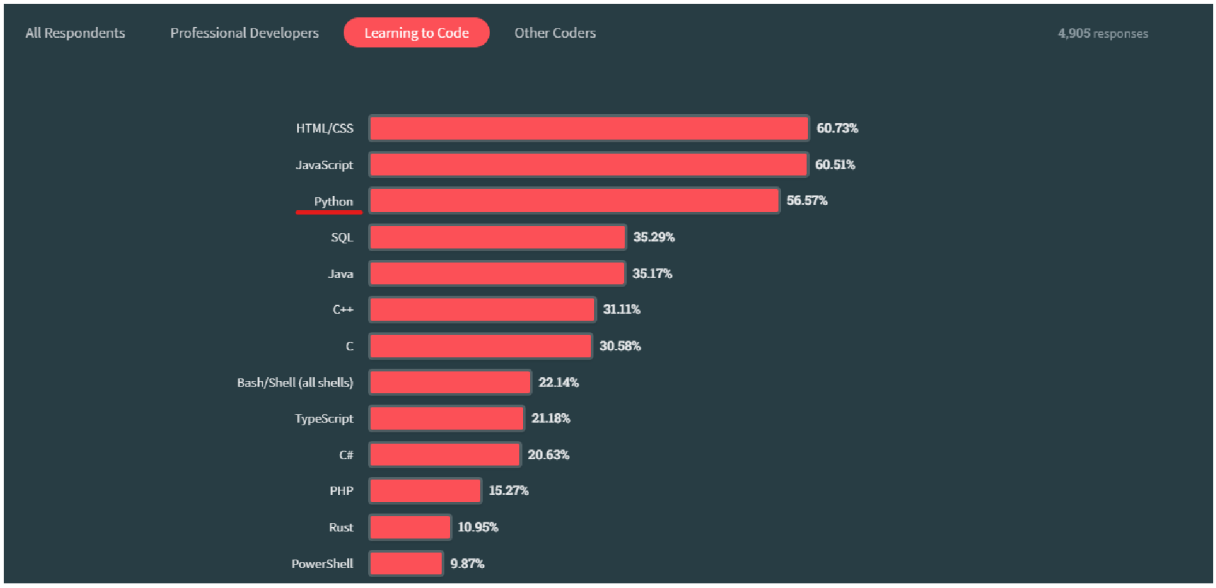
"Zlepši předchozí skript tak, aby funkce deleni kromě textu 'Chyba: dělení nulou' vrátila také hodnotu None pro případ dělení nulou."

Efektivní psaní promptů je důležité pro dosažení kvalitních odpovědí od AI nástrojů ať už při zadávání programovacích úloh, nebo v jakékoli jiné oblasti. Dodržováním doporučení jako jasnost a konkrétnost, použití jednoduchého jazyka, poskytování kontextu, uvádění příkladů vstupu a výstupu a zohlednění potenciálních chyb a omezení můžete výrazně zvýšit pravděpodobnost, že AI poskytne správnou a užitečnou odpověď.

2 Praktická část

Praktická část této práce se zaměřuje na posouzení využití AI nástrojů při výuce programování pro žáky druhého stupně základních škol. V této části budeme používat AI komunikující nástroj ChatGPT 4o. Tato aplikace je k dispozici zdarma. Pro použití ve výuce může být jistým omezením, že ChatGPT mohou používat děti až od 13 let. Alternativou proto může být například nástroj Codebreaker chatbot, který se specializuje pouze na programování a mohou ho používat bez svolení rodičů i děti pod 13 let (Národní pedagogický institut, 2023). Programovací úlohy budou postupně narůstat na obtížnosti a inspiraci k zadání čerpám z populární publikace "The Python Workbook" (Stephenson, 2014).

Pro výuku programování jsem zvolil dva programovací jazyky: Python a Scratch. Python je široce uznávaný jako ideální první programovací jazyk díky své jednoduché a čitelné syntaxi, která umožňuje snadné pochopení základních programovacích konceptů. Python je také velmi populární v různých oblastech od webového vývoje až po datovou analýzu, což z něj činí cenný nástroj nejen pro začátečníky, ale i pro pokročilé programátory (Guo, 2014). Data z průzkumu Stack Overflow Developer Survey 2023 ukazují, že Python je jedním z nejpopulárnějších programovacích jazyků jak mezi vývojáři, tak mezi lidmi, kteří se teprve programování učí, což potvrzuje jeho relevanci ve vzdělávání.



Obrázek 1 Nejpopulárnější programovací jazyky

(Zdroj: https://survey.stackoverflow.co/2023/#most-popular-technologies-language-learn)

Scratch, na druhé straně, je vizuální programovací jazyk, který byl speciálně navržen pro děti a začátečníky. Využívá blokové programování, které umožňuje uživatelům sestavovat kód pomocí grafických bloků místo psaní textového kódu. Tento přístup snižuje kognitivní zátěž spojenou s učením se syntaxi a umožňuje studentům zaměřit se na logiku a strukturu programování. Scratch je široce využíván ve vzdělávání a má silnou podporu komunity, což z něj činí ideální nástroj pro výuku programování na základních školách (Resnick et al., 2009).

Výběr AI nástroje ChatGPT 4o pro zadávání programovacích úloh byl motivován jeho schopností poskytovat interaktivní a adaptivní podporu studentům. ChatGPT 4o, vyvinutý společností OpenAI, je jedním z nejpokročilejších jazykových modelů dostupných v současnosti. Je schopen rozumět přirozenému jazyku a generovat koherentní odpovědi, díky čemuž je to dobrý adept na to, být efektivní nástroj pro interaktivní výuku programování. ChatGPT 4o může studentům poskytovat okamžitou zpětnou vazbu a pomáhat jim řešit problémy v reálném čase, což zvyšuje jejich angažovanost a efektivitu učení (Brown et al., 2020).

V praktické části této práce se budeme u některých úloh ve spolupráci s mou sestřenicí Viktorkou snažit řešit programovací úlohy různé obtížnosti prostřednictvím ChatGPT 4o. Viktorka je žákyně druhého stupně základní školy, nadaná na matematiku, která v rámci školy již má zkušenost s vizuálním programovacím jazykem Scratch. Jednodušší úkoly ji zkusím nejprve nechat vyřešit samotnou a budu pozorovat, jakým způsobem probíhá interakce s chatbotem. Úlohy byly navrženy tak, aby pokrývaly základní programovací koncepty, jako jsou proměnné, podmínky, cykly, funkce a rekurze. U obtížnějších úloh se budu ChatGPT 4o dotazovat na doplňující otázky. Cílem je zjistit, jak efektivně tyto AI nástroje podporují výuku programování, jaký je jejich dopad na učení a zhodnotit, jakým způsobem bude do budoucna vhodné tyto nástroje využívat.

Tímto způsobem se budu snažit zhodnotit didaktický potenciál AI nástroje ChatGPT 4o v kontextu výuky programování. Výsledky této analýzy by měly přispět k lepšímu pochopení toho, jak mohou AI nástroje podporovat vzdělávání v programování a jaké jsou jejich výhody a omezení. Pro čitelnost budu dále označovat nástroj ChatGPT 4o jako chatbot.

2.1 Vývoj vzorových programů

Na začátku každého oddílu vymezeného jedné programovací úloze bude specifikováno zadání. Následně je popsán průběh tvorby programů. Vzhledem k tomu, že velmi brzy začalo být jasné, že kód v Pythonu zvládne chatbot vyrobit zcela bez problémů, vygenerované kódy v Pythonu umístím až do příloh. Námi vytvořený program ve Scratchi na základně instrukcí z chatbotu bude vždy přiložen jako obrázek.

2.1.1 Pozdrav

Zadání:

Napiš program, který požádá uživatele, aby zadal své jméno. Program by měl odpovědět zprávou, která uživatele pozdraví a použije jeho jméno.

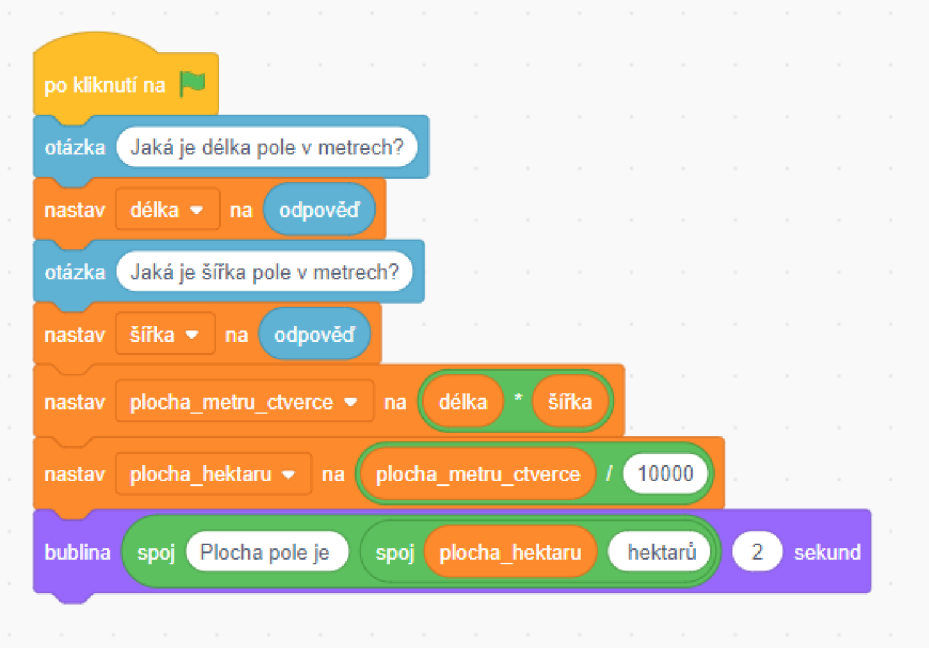
Zatímco v Pythonu chatbot vytvořil zcela správný, spustitelný a pěkně okomentovaný kód, u Scratche jsme i u takto jednoduchého programu narazili na problémy. Jistý problém je v lokalizaci do češtiny. ChatGPT nás vybízí k použití bloku „řekni…“, který v české verzi Scratche najdeme pod slovem „bublina“. Následně nás ChatGPT vybízí k výběru bloku ze sekce „Snímání“, což opět neodpovídá české lokalizaci, ve které je takto označená zřejmě sekce „Vnímání“. Máme použít blok „zeptat se“, který opět nenajdeme, jelikož je v české verzi tento blok označen „otázka“. V dalším kroku nám zase doporučuje chatbot nastavit hodnotu proměnné dříve, než jsme ji vůbec vytvořili.

Na jiná jména bloků zkusíme chatbot upozornit a uvidíme, jestli bude v dalších úkolech používat správná jména.

Napsal jsem prompt:

```
Děkuji. Chtěl bych jen upozornit na nedokonalosti s českou
lokalizací Scratche. Sekce Snímání ve Scratchi není, jmenuje se
totiž Vnímání. Příkaz "řekni" se jmenuje "bublina" a příkaz
"zeptat se", se jmenuje "otázka".
```

Po upozornění chatbot opravil instrukce tak, že už odpovídají české lokalizaci, nicméně i u takto jednoduchého zadání bylo potřeba se nad řešením trochu zamyslet a například použít spojovník “spoj”, ke kterému nás chatbot ani nevybídl.



Obrázek 2: Úkol „Plocha pole“ ve Scratchi

Zdroj: vlastní zpracování

2.1.2 Plocha pole

Zadání:

Vytvoř program, který přečte délku a šířku farmářského pole od uživatele v metrech. Zobrazí plochu pole v hektarech.

Tento úkol již vyžaduje práci s proměnnými a základní aritmetiku. V pythonu opět žádný problém, vznikl pěkně okomentovaný korektní, spustitelný kód, který splňuje všechny naše požadavky. Po implementaci našich upozornění na nedokonalosti v lokalizaci už jsme dostali od chatbotu přesnější instrukce a ke konstrukci spustitelného řešení nás chatbot dovedl téměř bez zádrhele.

2.1.3 Výpočet BMI

Zadání:

Napiš program, který vypočítá index tělesné hmotnosti (BMI) jednotlivce. Tvůj program by měl začít tím, že přečte výšku a váhu od uživatele. Poté by měl použít následující vzorec pro výpočet BMI a zobrazit výsledek. Pokud bude výška zadaná v metrech a váha v kilogramech, pak je index tělesné hmotnosti vypočítán podle tohoto vzorce:

BMI = váha / (výška × výška).

Při spuštění kódu vygenerovaného chatbotem narazila má sestřenice na problém. Zadala totiž svou výšku pomocí české notace používající čárku pro oddělení desetinných míst. Program proto skončil předčasně. Využil jsem této příležitosti a dotázali jsme se chatbotu, kde nastal problém? Vždy je ideální přímo zkopírovat chybovou hlášku programu. To jsme udělali a chatbot velmi srozumitelně vysvětlil, kde nastal problém. Tento úkol byl s použitím instrukcí z chatbotu jednoduše zvládnutelný za krátký čas.

Obrázek 4: Úkol „Výpočet BMI“ ve Scratchi

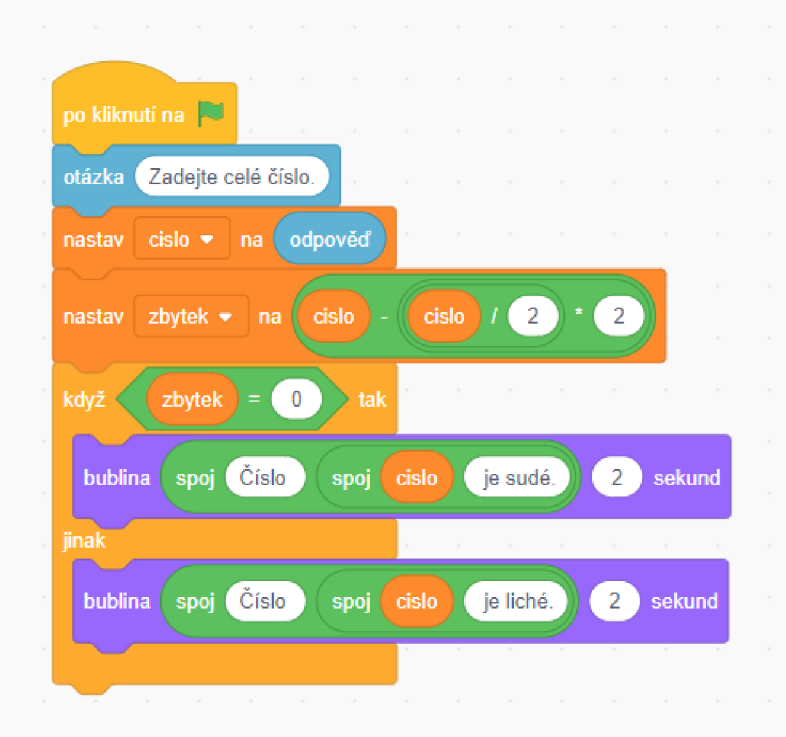
Zdroj: vlastní zpracování

2.1.4 Je číslo sudé nebo liché?

Zadání:

Napiš program, který přečte celé číslo od uživatele. Poté tvůj program zobrazí zprávu, která indikuje, zda je celé číslo sudé nebo liché.

V tomto úkolu jsme se sestřenicí narazili na větší nedorozumění co se týče instrukcí pro Scratch. S kódem v Pythonu si AI opět poradila bez problémů, narozdíl od Scratche. Chatbot totiž automaticky v instrukcích používal operátor modulo, který ovšem ve Scratchi pod tímto názvem nenajdeme. Nechal jsem tedy sestřenici chvíli přemýšlet, jak by k výsledku mohla dojít vybízejíce ke konzultaci AI. Upozornili jsme tedy chatbot, že nemůžeme ve Scratchi najít operátor modulo. Chatbot nám proto nabídl řešení, které operátor modulo nepoužívá. Jenže toto řešení není správné a pro všechny hodnoty proměnné by číslo vyšlo jako sudé. Chatbot zcela opomněl fakt, že operátor modulo se ve Scratchi nachází, ale pod jiným názvem. Tento úkol byl proto příklad, při kterém konzultace s chatbotem způsobovala spíše další zmatení, než cestu ke správnému řešení.



Obrázek 5: Úkol „Je číslo sudé nebo liché?“ ve Scratchi

Zdroj: vlastní zpracování

2.1.5 Roční období dle data

Zadání:

Rok je rozdělen do čtyř ročních období: jaro, léto, podzim a zima. Přestože se přesné data, kdy se roční období mění, mohou mírně lišit rok od roku kvůli způsobu, jakým je kalendář sestaven, pro tento úkol použijeme následující data:

Jaro: 20. března

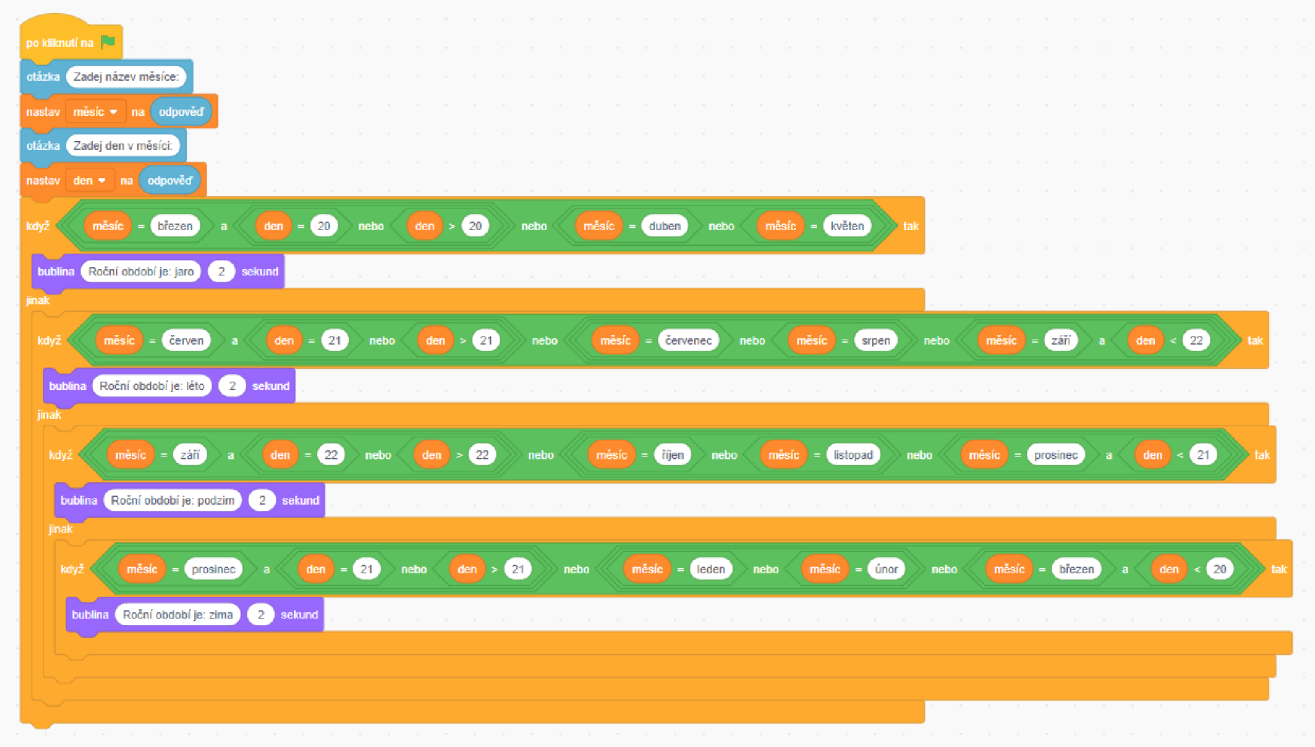
Léto: 21. června

Podzim: 22. září

Zima: 21. prosince

Vytvoř program, který přečte měsíc a den od uživatele. Uživatel zadá název měsíce jako řetězec, následovaný dnem v měsíci jako celé číslo. Poté tvůj program zobrazí roční období spojené se zadaným datem.

Tento úkol testuje složitější větvení podmínek. V Pythonu opět dospěl chatbot ke korektnímu, spustitelnému řešení. Dokonce ošetřil podmínku pro zadání neplatného formátu data.



Obrázek 6: Úkol „Roční období dle data“ ve Scratchi

Zdroj: vlastní zpracování

U Scratche začalo být po chvíli snahy o vytvoření kódu mou sestřenicí zjevné, že pro náročnější úkoly pracující s textem není Scratch ideální nástroj. Booleovské operátory (AND, OR, NOT) nejdou ve Scratchi řetězit, ale pouze vkládat do sebe, což zhoršuje přehlednost kódu. Zároveň ve Scratchi nenajdeme operátory menší nebo rovno, větší nebo rovno, a tak pokud

bychom následovali přesně instrukce z chatbotu, vznikne lehce neúsporné řešení, které ale po spuštění pracuje správně. Oproti Pythonu také chatbot neošetřil podmínkou neplatné vstupy.

2.1.6 Je řetězec palindrom?

Zadání:

Řetězec je palindrom, pokud je stejný při čtení dopředu i dozadu. Například "anna", "kajak", "radar" a "nepotopen" jsou všechny příklady palindromických slov. Napiš program, který přečte řetězec od uživatele a pomocí cyklu určí, zda je nebo není palindrom. Zobraz výsledek, včetně smysluplné výstupní zprávy.

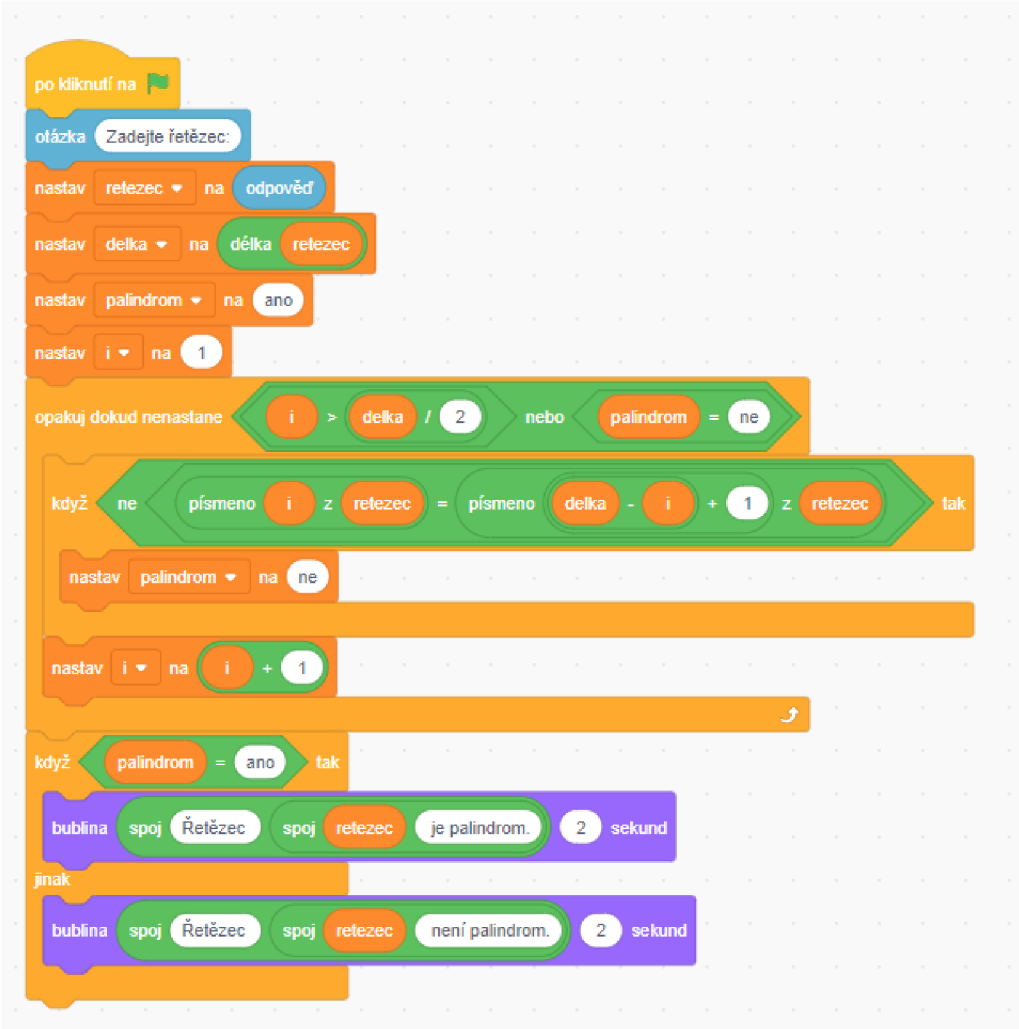
S přibývající složitostí úkolů začínám sledovat jisté vzorce. Kód v pythonu chatbot vytvořil bez problému a se všemi náležitostmi dobrého kódu. Nechal jsem moji sestřenici zkusit vytvořit program ve Scratchi dle instrukcí. Tentokrát opět bez většího úspechu. První problém nastal, když nás chatbot vyzval k použití cyklu “Opakuj dokud…”, který ve Scratchi neexistuje, najdeme v něm totiž jen cyklus “Opakuj dokud nenastane”.Jedná se o Scratch verzi while cyklu, ovšem s opačnou podmínkou. Upozornili jsme proto na tuto nesrovnalost chatbot, a ten řešení upravil. Přestože pro zkušenějšího uživatele Scratche může být rozdíl snadno pochopitelný, pro sestřenici nejasnost v příkazech způsobila značné zmatení. Pokračujeme dále. Bloky se tedy podařilo přibližně nastavit podle instrukcí, ale dostali jsme se k dalšímu problému. Chatbot vyjádřil podmínku jako delka - i + 1, což je jednoduchá aritmetika, nicméně ve Scratchi nemohu operátory sčítání takto jednoduše řetězit.

Všechny operátory jsou totiž ve Scratchi binární a řetězení mohu docílit pouze vnořením do sebe, což splňuje zároveň stejnou funkci jako uzávorkování. Proto je třeba věnovat velkou pozornost pořadí operací.

Obrázek 7: Podobný vzhled, rozdílný význam operací ve Scratchi

Zdroj: vlastní zpracování

Tyto dva výrazy viz obr.7 mají rozdílný význam, což ovšem můžeme snadno přehlédnout, což se nám i stalo. Oslovili jsme proto znovu chatbota, aby nám pomohl, kde můžeme mít chybu s tím, že tušíme nesprávně zadané podmínky. Chatbot ale na chybu nepřišel a vyzval nás k ujištění, že zadáváme řetězce bez speciálních znaků či mezer. Po úpravách podmínek ale program pracuje správně.



Obrázek 8: Úkol „Je řetězec palindrom?“ ve Scratchi

Zdroj: vlastní zpracování

2.1.7 Binární číslo na desítkové

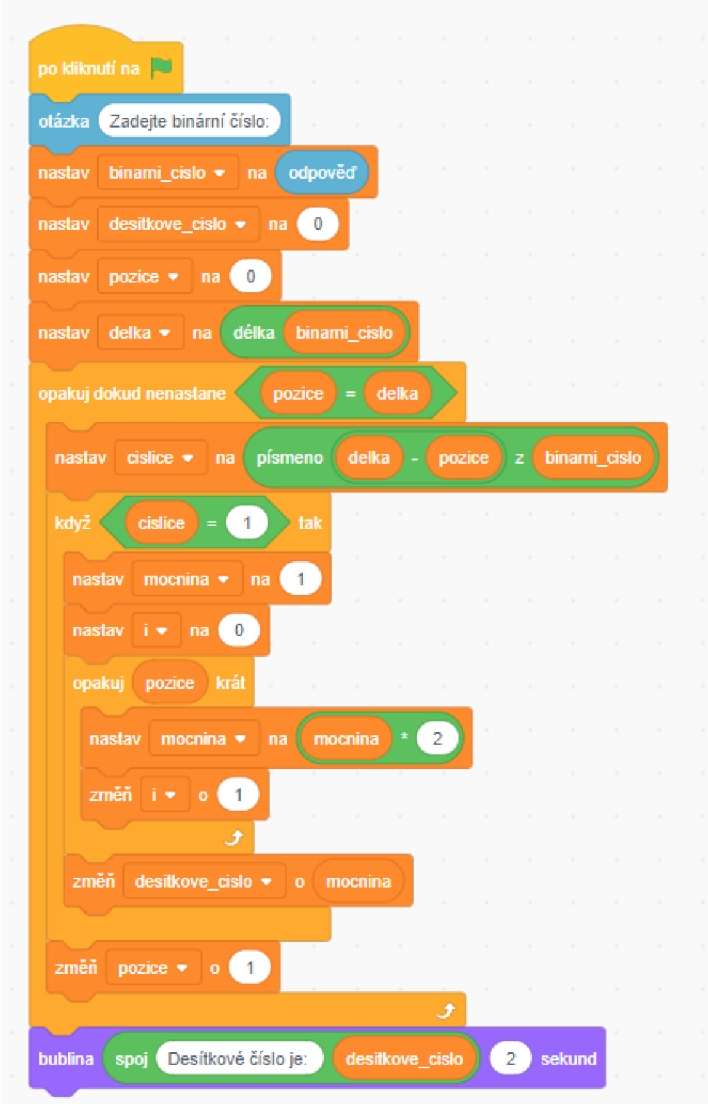
Zadání:

Napiš program, který převede binární číslo na číslo v desítkové soustavě. Tvůj program by měl začít tím, že přečte binární číslo od uživatele jako řetězec. Poté by měl spočítat

ekvivalentní desítkové číslo zpracováním každé číslice v binárním čísle. Nakonec by tvůj program měl zobrazit ekvivalentní desítkové číslo s odpovídající zprávou.

Tento úkol byl pro moji sestřenici už trochu příliš obtížný, a tak jsem se do něj pustil sám a výsledky mě téměř šokovaly. Ale pojďme popořádku. V Pythonu vygeneroval chatbot jako je už zvykem zcela správný program. V první verzi ale použil některé speciální funkce jako enumerate, tak jsem chatbot požádal ještě o verzi se základními funkcemi pro jednodušší pochopení algoritmu. Přišel i s vysvětlením postupu.

Následně jsem se dostal k implementaci instrukcí od chatbota ve Scratchi. Prvotní verze obsahovala ve Scratchi neexistující mocninný operátor, který se dá nahradit jedině cyklem. Chatbota jsem na to upozornil a bez problémů přišel s řešením bez mocninného operátoru, ale s cyklem. Když jsem ale přesně následoval instrukce, program nefungoval správně. Pro malé hodnoty do 4 pozic binárního čísla vracel desítkové číslo o 1 menší a u větších čísel vracel již zcela nesprávné hodnoty. V této chvíli už jsem začal být skeptický, že si s úkolem chatbot poradí a očekával jsem, že ke správnému řešení dojdu už jedině svépomocí. Jenže když jsem chatbotu napsal, jaké nesprávné výsledky z programu dostávám, opravil jedno zanořené nastavování proměnné a nová verze programu, pozměněná pouze v jednom řádku, již funguje zcela správně. Když jsem tedy dobře popsal, jaké nesprávné výsledky dostávám, chatbot byl schopen správně reagovat.



Obrázek 9: Úkol „Binární číslo na desítkové“ ve Scratchi

Zdroj: vlastní zpracování

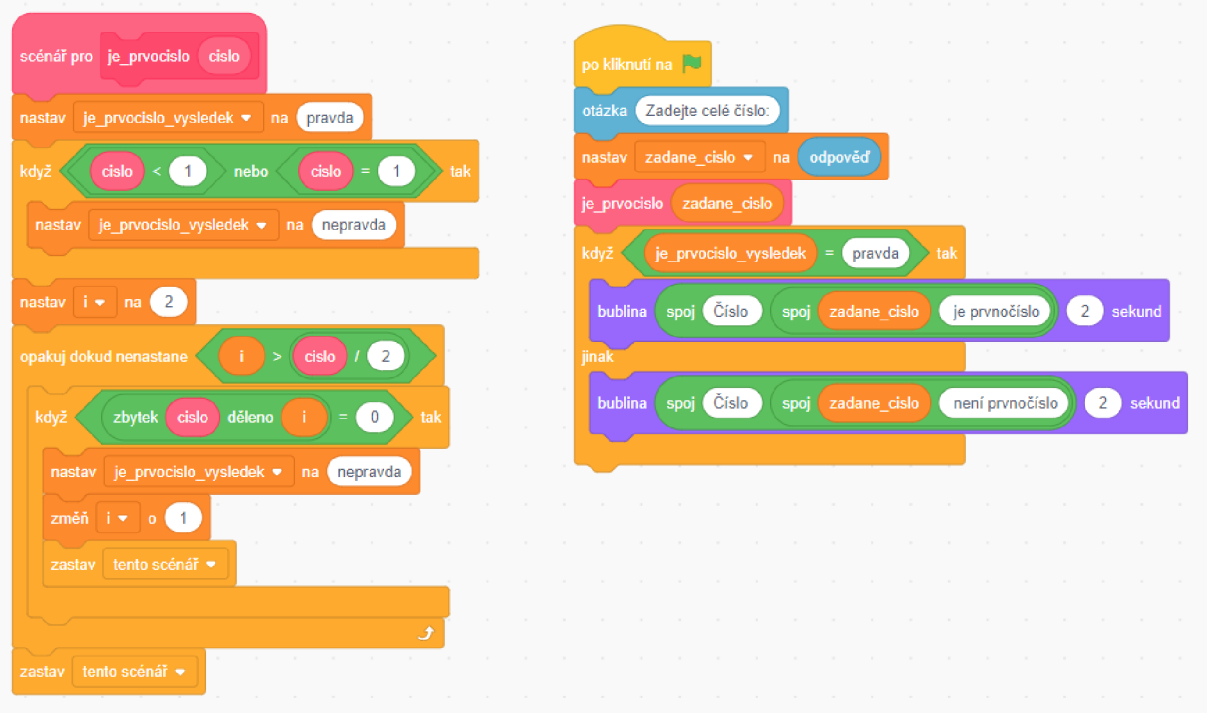
2.1.8 Je číslo prvočíslo?

Zadání:

Prvočíslo je celé číslo větší než 1, které je dělitelné pouze jedničkou a samo sebou. Napiš funkci, která určí, zda je její parametr prvočíslo, vrátí True, pokud je, a False, pokud není. Napiš hlavní program, který přečte celé číslo od uživatele a zobrazí zprávu, která indikuje, zda je nebo není prvočíslo.

Tento úkol cílí především na to, rozdělit si program do funkcí. Má sestřenice ještě nebyla seznámena s možností volání různých scénářů, a tak jsem program implementoval téměř bez jejího zásahu. V případě Pythonu nedošlo k žádnému překvapení, chatbot si s úkolem poradil zcela bez problémů a vytvořil spustitelný kód, který vrací správné hodnoty.

Zajímavější úkol to proto bude u Scratche, kde volání funkcí, neboli Scénářů patří spíše už k pokročilým možnostem tohoto vizuálního jazyka. Náš požadavek k rozdělení programu do funkcí chatbot nijak nerozhodil a když jsem vytvořil program na základě zadaných instrukcí, blížil jsem se správnému řešení, nicméně volaná funkce nikdy neukončila svůj průběh, a tak jsme se nikdy nedostali k definitivnímu výsledku. Na tento neduh jsem chatbot upozornil. Navrhl použití bloku "stop", který všem v české lokalizaci najdeme pod jménem "zastav". Chatbot sice tyto bloky do kódu vložil, ale na nesprávné místo a proto bylo třeba algoritmus ještě trochu změnit. I s lehkým klopýtnutím ale chatbot dodal velmi koherentní instrukce, které vedly k cíli.



Obrázek 10: Úkol „Je číslo prvočíslo?" ve Scratchi

Zdroj: vlastní zpracování

2.1.9 Palindrom rekurentně

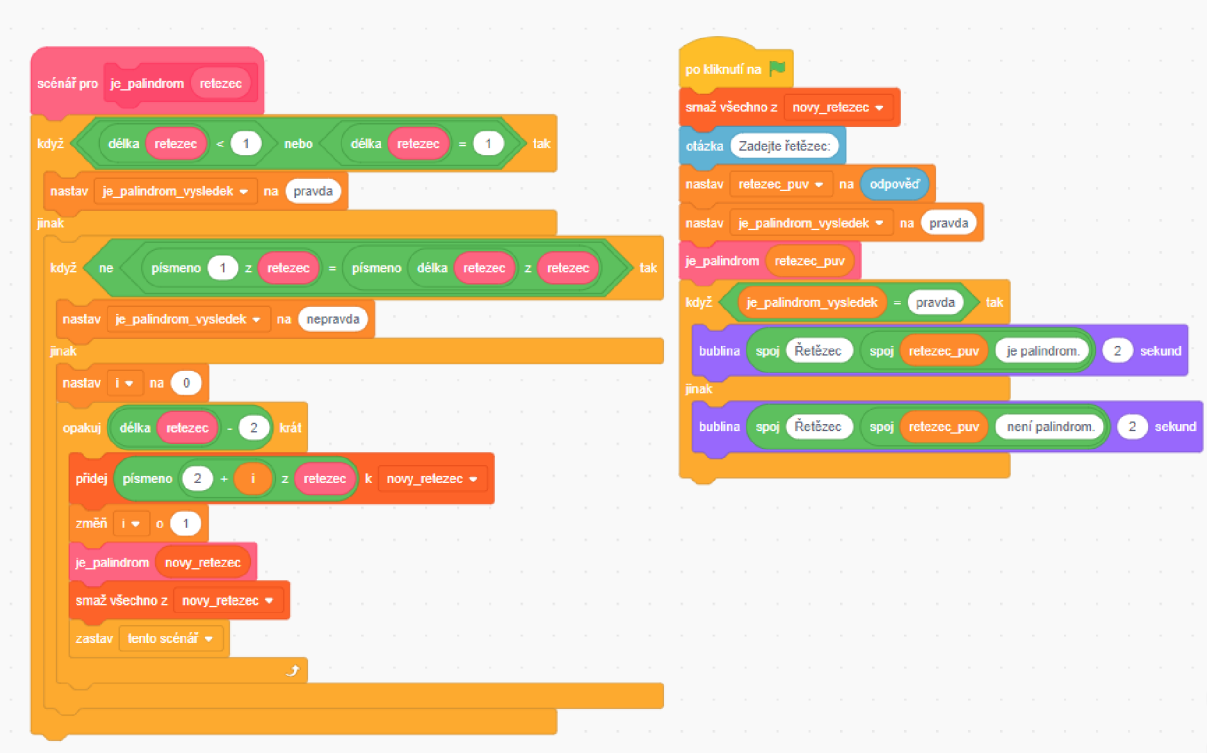
V tomto úkolu vyzkoušíme možnosti rekurze a použití polí.

Zadání:

Koncept palindromu byl představen dříve. V tomto cvičení napíšete rekurzivní funkci, která určí, zda je řetězec palindrom. Prázdný řetězec je palindrom, stejně jako každý řetězec obsahující pouze jeden znak. Jakýkoli delší řetězec je palindrom, pokud jeho první a poslední znaky se shodují a pokud řetězec vytvořený odstraněním prvního a posledního znaku je také palindrom.

Napište hlavní program, který přečte řetězec od uživatele. Použijte svou rekurzivní funkci k určení, zda je řetězec palindrom. Poté zobrazte odpovídající zprávu pro uživatele.

Co se týče navrhovaného řešení úkolu chatbotem v Pythonu, opět příliš není co rozebírat. S úkolem si poradil bez problémů a výstup srozumitelně okomentoval. Pracoval s pythonovskou notací, která umožňuje pracovat s řetězci pomocí přístupu k jednotlivým členům a s vytvořením nového, zkráceného řetězce.



Obrázek 11: Úkol „Palindrom rekurentně“ ve Scratchi

Zdroj: vlastní zpracování

Řešení ve Scratchi představuje větší výzvu. Chatbot se v průběhu vrátil k pojmenovávání některých bloků jiným názvem. První iterace programu ale poskytuje slušnou kostru, která se dá rozvíjet. Zprvu nás chatbot vybízí k použití příkazů, jako

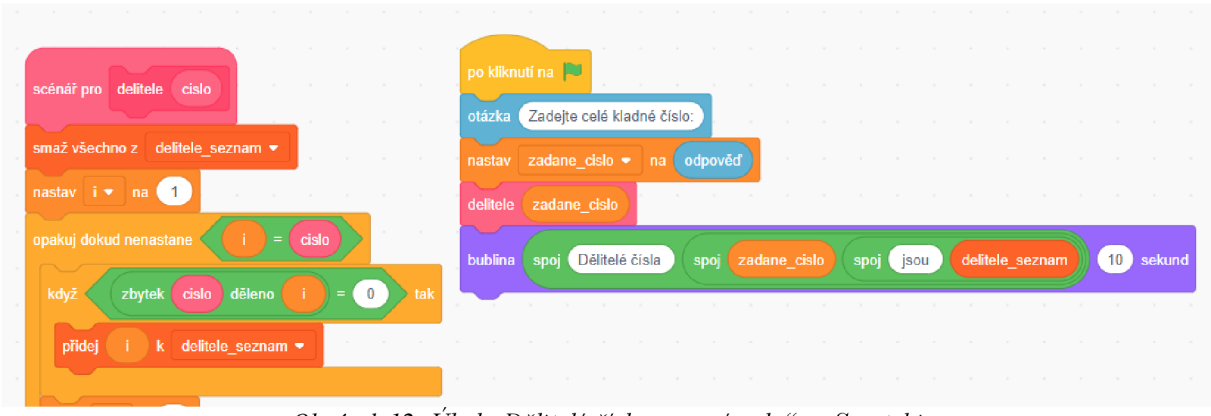
```
nastav [novy_retezec] na (podřetězec (retezec) od #2 do #(délka (retezec) - 1))
```

, jenomže takovými operacemi Scratch nedisponuje. Po upozornění chatbot reaguje a navrhuje problém vyřešit vytvořením nového řetězce po jednom znaku. Z kontextu jsem pochopil, že nás chatbot vybízí k použití datové struktury seznam, který nepatří mezi nejpoužívanější ve Scratchi a v základu jsou funkce spojené se seznamy ve Scratchi skryty. Chatbot termíny seznam a řetězec zaměňuje. I po upozornění také není schopen ošetřit vyčištění dat ze seznamu po opakovaném spuštění, což je překážka, která způsobuje chybné výsledky. Musím ale ocenit téměř funkční kostru programu. Sice byly nutné dodatečné zásahy k ošetření správné funkcionality, ale struktura byla nosná.

2.1.10 Dělitelé čísla pomocí pole

Zadání:

Dělitel kladného celého čísla n je kladné celé číslo menší než n, které n dělí beze zbytku. Napiš funkci, která vypočítá všechny dělitele kladného celého čísla. Číslo bude předáno funkci jako jediný parametr. Funkce vrátí seznam obsahující všechny správné dělitele jako svůj jediný výsledek. Dokonči toto cvičení napsáním hlavního programu, který demonstruje funkci tím, že přečte hodnotu od uživatele a zobrazí seznam jejích správných dělitelů.



Obrázek 12: Úkol „Dělitelé čísla pomocí pole“ ve Scratchi

Zdroj: vlastní zpracování

Tento úkol se zaměřuje na základní použití funkcí a polí (seznamů). V případě Pythonu chatbot opět vytvořil správně formátovaný, úsporný a okomentovaný kód, a tak se zaměříme na tvorbu kódu dle instrukcí ve Scratchi. V tomto případě chatbot ani ve Scratchi nijak nezaváhal. Při následování přesných instrukcí jsme se dostali ke správnému výsledku. Oproti předchozím příkladům práce se seznamy se chatbot dokonce poučil a smazal předchozí seznam na začátku volání funkce, která vytváří seznam nový. Scratch totiž při opakovaných spuštěních programu uchovává hodnoty v seznamu nezměněné, pokud hodnoty nevymažeme příkazem.

2.2 Analýza a porovnání programů

V této sekci se pokusím o souhrnnou analýzu toho, jakým způsobem tvorba programů a komunikace s chatbotem probíhala. V první řadě musím vyzdvihnout velmi intuitivní ovládání, zprostředkované prostředím, které se podobá klasickým chat aplikacím jako Messenger nebo WhatsApp. V teoretické části práce jsme si vymezili vlastnosti dobrého kódu, a pokud jde o programy v Pythonu vygenerované AI, vždy přišla s naprosto bezchybným řešením a pokud bylo třeba něco dodatečně specifikovat, výstup AI korektně opravila. Pokud je tedy úkol dobře formulovaný a podobného rozsahu a obtížnosti jako úkoly v této práci, v současné verzi Chat GPT 4o je výstup v Pythonu nedokonalý jen raritně.

Zajímavější věci se děly při snaze vytvořit úkol ve Scratchi. V současné verzi není ChatGPT schopen přímo vytvořit program v prostředí vizuálního programovacího jazyka, a tak vznikalo spousta nesrovnalostí či chyb. Chatbot nám vygeneruje postup jak naskládat programovací bloky tak, aby vznikl program dle zadání. První problém nastal v lokalizaci. ChatGPT 4o pracuje primárně s jazykovým modelem v angličtině a následně výstup překládá do češtiny. Proto vznikaly nesrovnalosti v názvech sekcí bloků nebo operací. Zkušeného uživatele Scratche to nejspíš nemusí rozhodit, ale má pozorovací žákyně se musela nad tímto silně zamýšlet. Rozdílnost názvů bloků je ale řekl bych menší problém. V některých případech si totiž chatbot vyloženě vymýšlel některé ve Scratchi neexistující konstrukty. Vytvoření programu ve Scratchi bylo proto mnohem náročnější a častěji byla nutná další úprava. Nutno ale zmínit, že v některých případech mě schopnost chatbotu reagovat na mé připomínky ohromila. Především v příkladu číslo 7. Již vcelku rozsáhlý program nefungoval správně a po upozornění na nesprávný výstup byl schopen rozeznat chybu v jednom bloku a sám ji opravit. Další problém představovalo řetězení operací, jelikož Scratch vždy umožňuje jen binární

operace a řetězení těchto operací je realizování zanořováním, což nebyl chatbot schopen reflektovat. Obecně nás ale vždy chatbot minimálně navedl na správné řešení.

2.3 Zhodnocení výukového potenciálu

V předchozí podkapitole jsem zhodnotil, jak si chatbot poradil s typickými příklady programovacích úloh. V této podkapitole na to ale nahlédneme ze strany výukového potenciálu. Moje sestřenice se s prostředím Chatu GPT sžila velmi rychle. Možnost zcela individuální, okamžité zpětné vazby je určitě skvěle využitelný nástroj, obzvlášť v disciplínách, které si vyžadují větší pozornost a čas, jako je například programování. Práce na programech v Pythonu a ve Scratchi byly ale dva velmi rozdílné zážitky, pokud jde o výukový potenciál. Úkoly v Pythonu byl totiž chatbot schopen realizovat skvěle bez žádných problémů, což vybízí k zamyšlení, jaký může mít vůbec taková spolupráce přínos. Pokud je člověk silně vnitřně motivován a chce srozumitelně pochopit koncepty, chatbot je nenahraditelný společník. Mohu ho požádat o dodatečné vysvětlení, další příklady a s každým tématem může strávit libovolně času a prostoru. Silná vnitřní motivace se něco naučit ale dle mé zkušenosti nemusí být typická vlastnost žáka druhého stupně ZŠ. Možnost nechat úkol vyřešit chatbot za mě bude pro žáky příliš lákavá na to, aby to bez stanovení nových pravidel nedělali. V tomto případě nastává problém, že chatbot jakýkoli úkol, který by v současné době realisticky byl požadován po žáku druhého stupně ZŠ, bez problémů vyřeší. A tak bude dle mého názoru potřeba zcela změnit paradigma, jak na takovou výuku nahlížíme. Zadat žákům za domácí úkol několik příkladů, na kterých si procvičí programovací koncepty a následně kontrolovat jen výsledek, zřejmě ztrácí smysl. Minimálně v programovacím jazyku Python.

Zcela jiný příběh jsem pozoroval s vývojem programů ve Scratchi. Skutečnost, že v současné době není chatbot schopen přímo vytvořit takový program a navíc vcelku často uvádí nepřesné instrukce, znamená, že paradoxně spolupráce s ním může mít větší výukovou hodnotu. To je i můj dojem z pozorování práce mé sestřenice. Byla nucena prozkoumat jednotlivé instrukce, následovat přesně algoritmus a vytvoření programu vyžadovalo i jakési vlastní objevování a invenci. Pokud nastal zádrhel, mohu se chatbotu dotázat a ten mě navede na řešení, které ale nebývá dokonalé.

Napadá mě proto několik změn náhledu na současnou výuku programování ve světě AI. Myslím si, že nástroje AI bude dobré limitovat pro žáky tak, aby více připomínaly učitele. Tedy aby měli děti do určitého věku přístup pouze k takovým nástrojům, které nechávají prostor pro učení, objevování a nevygenerují okamžitě výsledek. Jiný přístup by mohl být v zaměření na

vysvětlování konceptů a trávení více času v reálném světě. V takovém případě by pro žáka nebyl hlavním úkolem kus kódu, ale například prezentování jednotlivých programovacích konceptů, více diskuze a podobných metod. Dalo by se také polemizovat o tom, že bychom mohli jako učitelé vyžadovat řešení složitějších problémů, které by byly už příliš obtížné i pro AI, ale obávám se, že bychom velmi brzy narazili na limity lidského těla a naopak stálý růst schopností AI.

Závěr

V rámci teoretické části jsem vymezil hlavní pojmy související s využíváním AI v rámci výuky programování na základních školách. Bylo uvedeno několik definic AI a také její historie v kontextu vzdělávání. Následně jsme s pomocí relevantních zdrojů charakterizovali výhody, nevýhody a příležitosti užívání AI ve výuce. Zmínil jsem různé metody a nástroje, které mohou být důležité v době AI. Věnovali jsme se také specifikům výuky programování a provázaností s AI nástroji.

Stěžejní částí práce byla praktická část, ve které jsem na základě AI vygenerovaných programovacích úloh analyzoval výukový potenciál využití těchto nástrojů. Pro mne hlavním východiskem je, že je to nástroj ohromně nápomocný, ale ve vztahu k didaktickému potenciálu bude třeba výrazně změnit paradigma, jak vůbec o výuce programování přemýšlíme. Programy v programovacím jazyce Python totiž AI chatbot vygeneruje velmi kvalitně téměř bez výjimek. Na druhou stranu, ve vizuálním programovacím jazyce Scratch zatím AI není schopna vygenerovat výsledek perfektně a velká část práce zůstává na uživateli. Z didaktického hlediska proto vidím větší potenciál v interakci se Scratchem. Na základě zkušeností z průběhu práce jsem tedy zhodnotil výukový potenciál těchto nástrojů a nastínil scénáře, ve kterých bude interakce žáka s AI ve výuce programování vhodná. Cíle práce byly tedy za mě splněny.

Seznam použitých zdrojů

Literatura

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. The Journal of the Learning Sciences, 4(2), 167-207.

Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., & Palincsar, A. (1991). Motivating project-based learning: Sustaining the doing, supporting the learning. Educational Psychologist, 26(3-4), 369-398.

Calo, R. (2017). Artificial intelligence policy: A primer and roadmap. UCLA Law Review, 51(4), 399-435.

Carbonell, J. R. (1970). AI in CAI: An artificial-intelligence approach to computer-assisted instruction. IEEE Transactions on Man-Machine Systems, 11(4), 190-202.

Dillenbourg, P. (1999). What do you mean by collaborative learning? In P. Dillenbourg (Ed.), Collaborative-learning: Cognitive and Computational Approaches (pp. 1-19). Oxford: Elsevier. ISBN 9780080433179.

Fletcher, J. D., & Atkinson, R. K. (2015). Computer-based adaptive learning: A review of the literature. Journal of Educational Technology Systems, 43(3), 237-267.

Frey, C. B., & Osborne, M. A. (2017). The future of employment: How susceptible are jobs to computerization? Technological Forecasting and Social Change, 114, 254-280.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. Advances in Neural Information Processing Systems, 27.

Guo, P. J. (2014). Python is now the most popular introductory teaching language at top US universities. Communications of the ACM, 57(10), 12-13.

Holmes, W., Bialik, M., & Fadel, C. (2019). Artificial Intelligence in Education: Promises and Implications for Teaching and Learning. Center for Curriculum Redesign. ISBN 9781791218606.

Hunt, A., & Thomas, D. (1999). The Pragmatic Programmer: Your Journey to Mastery. Addison-Wesley Professional. ISBN 9780201616224.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255-260.

Luckin, R., Holmes, W., Griffiths, M., & Forcier, L. B. (2016). Intelligence Unleashed: An Argument for AI in Education. Pearson Education. ISBN 9781292163660.

Luxton-Reilly, A. (2016). The importance of automatic assessment in introductory programming courses. ACM Inroads.

McConnell, S. (2004). Code Complete: A Practical Handbook of Software Construction. Microsoft Press. ISBN 9780735619678.

O'Neil, C. (2016). Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy. Crown Publishing Group. ISBN 9780553418811.

Piaget, J. (1972). The Principles of Genetic Epistemology. London: Routledge & Kegan Paul. ISBN 9780710065296.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). Scratch: programming for all. Communications of the ACM, 52(11), 60-67.

Russell, S., & Norvig, P. (2016). Artificial Intelligence: A Modern Approach. Pearson Education. ISBN 9780134610993.

Selwyn, N. (2019). Should robots replace teachers? AI and the future of education. Digital Education Review, (35), 105-117.

Singh, R., Barry, B., Mott, B., & Lester, J. (2019). Gradescope: Automated AI grading system for large-scale online education. International Journal of Artificial Intelligence in Education, 29(3), 380-398.

Stephenson, B., 2019. *The Python Workbook*. 2nd ed. Cham: Springer. ISBN 9783030188733.

Vygotsky, L. S. (1978). Mind in Society: The Development of Higher Psychological Processes. Cambridge, MA: Harvard University Press. ISBN 9780674576292.

Wenger, E. (1987). Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge. Morgan Kaufmann. ISBN 9781558600246.
West, S. M. (2019). Data capitalism: Redefining the logics of surveillance and privacy. Business & Society, 58(1), 20-41.

Elektronické zdroje

AI100. (2023). *Defining AI. One Hundred Year Study on Artificial Intelligence*. Stanford University. Dostupné z: https://ai100.stanford.edu/

Caltech. (2023). *What Is AI?* Caltech Science Exchange. Dostupné z: https://scienceexchange.caltech.edu/topics/artificial-intelligence-research/artificial-intelligence-definition

Copeland, B. (2024, June 14). *artificial intelligence*. *Encyclopedia Britannica*. Dostupné z: https://www.britannica.com/technology/artificial-intelligence

Národní pedagogický institut. (2024). Generativní umělá inteligence pro učitele. [Video]. YouTube. Dostupné z: https://www.youtube.com/watch?v=As5j3p1sv5Q

Stack Overflow. (2023). Developer survey results 2023. Dostupné z: https://survey.stackoverflow.co/2023/

Stanford. (2023). What is AI? Stanford University. Dostupné z: https://hai.stanford.edu/sites/default/files/2023-03/AI-Key-Terms-Glossary-Definition.pdf

Seznam obrázků

Obrázek 1 Nejpopulárnější programovací jazyky 31
Obrázek 2: Úkol „Pozdrav“ ve Scratchi 34
Obrázek 3: Úkol „Plocha pole“ ve Scratchi 34
Obrázek 4: Úkol „Výpočet BMI“ ve Scratchi 35
Obrázek 5: Úkol „Je číslo sudé nebo liché?“ ve Scratchi 36
Obrázek 6: Úkol „Roční období dle data“ ve Scratchi 37
Obrázek 7: Podobný vzhled, rozdílný význam operací ve Scratchi 38
Obrázek 8: Úkol „Je řetězec palindrom?“ ve Scratchi 39
Obrázek 9: Úkol „Binární číslo na desítkové“ ve Scratchi 41
Obrázek 10: Úkol „Je číslo prvočíslo?“ ve Scratchi 42
Obrázek 11: Úkol „Palindrom rekurentně“ ve Scratchi 43
Obrázek 12: Úkol „Dělitelé čísla pomocí pole“ ve Scratchi 44

Seznam příloh

Příloha 1 – ChatGPT 4o vygenerované kódy v Pythonu

Příloha 1 – Chat GPT 4o vygenerované kódy v Pythonu

```
# Program, který požádá uživatele o zadání jména a pozdraví ho


# Požádáme uživatele o zadání jména
jmeno = input("Zadejte své jméno: ")


# Odpovíme zprávou s pozdravem
print("Ahoj, " + jmeno + "!")
```

```
# Program, který přečte délku a šířku farmářského pole a
zobrazí plochu v hektarech


# Požádáme uživatele o zadání délky a šířky pole v metrech
delka = float(input("Zadejte délku pole v metrech: "))
sirka = float(input("Zadejte šířku pole v metrech: "))


# Vypočítáme plochu pole v metrech čtverečních
plocha_metru_ctverece = delka * sirka


# Převedeme plochu z metrů čtverečních na hektary (1 hektar =
10,000 metrů čtverečních)
plocha_hektaru = plocha_metru_ctverece / 10000


# Zobrazíme plochu pole v hektarech
print("Plocha pole je:", plocha_hektaru, "hektarů")
```

```
# Program, který vypočítá index tělesné hmotnosti (BMI)

# Požádáme uživatele o zadání výšky v metrech a váhy v
kilogramech
vyska = float(input("Zadejte svou výšku v metrech: "))
vaha = float(input("Zadejte svou váhu v kilogramech: "))

# Vypočítáme BMI podle vzorce
bmi = vaha / (vyska * vyska)

# Zobrazíme BMI
print("Vaše BMI je:", bmi)
```

```
# Program, který určí, zda je zadané číslo sudé nebo liché

# Požádáme uživatele o zadání celého čísla
cislo = int(input("Zadejte celé číslo: "))

# Zjistíme, zda je číslo sudé nebo liché
if cislo % 2 == 0:
    print("Číslo", cislo, "je sudé.")
else:
    print("Číslo", cislo, "je liché.")
```

```
# Program, který určí roční období podle zadaného data


# Funkce pro určení ročního období
def urcit_obdobi(mesic, den):
    if (mesic == "březen" and den >= 20) or (mesic in
["duben", "květen"]) or (mesic == "červen" and den < 21):
        return "jaro"
    elif (mesic == "červen" and den >= 21) or (mesic in
["červenec", "srpen"]) or (mesic == "září" and den < 22):
        return "léto"
    elif (mesic == "září" and den >= 22) or (mesic in
["říjen", "listopad"]) or (mesic == "prosinec" and den < 21):
        return "podzim"
    elif (mesic == "prosinec" and den >= 21) or (mesic in
["leden", "únor"]) or (mesic == "březen" and den < 20):
        return "zima"
    else:
        return "Neplatné datum"


# Požádáme uživatele o zadání měsíce a dne
mesic = input("Zadejte název měsíce: ").lower()
den = int(input("Zadejte den v měsíci: "))


# Určíme roční období
obdobi = urcit_obdobi(mesic, den)


# Zobrazíme roční období
print("Roční období je:", obdobi)
```

```
# Program, který určí, zda je zadaný řetězec palindrom

# Požádáme uživatele o zadání řetězce
retezec = input("Zadejte řetězec: ")

# Inicializujeme proměnnou pro zpětný řetězec
retezec_zpetne = ""

# Použijeme cyklus k vytvoření zpětného řetězce
for znak in retezec:
    retezec_zpetne = znak + retezec_zpetne

# Porovnáme původní a zpětný řetězec
if retezec == retezec_zpetne:
    print(f"Řetězec '{retezec}' je palindrom.")
else:
    print(f"Řetězec '{retezec}' není palindrom.")
```

```
# Program, který převede binární číslo na číslo v desítkové soustavě

# Požádáme uživatele o zadání binárního čísla
binarni_cislo = input("Zadejte binární číslo: ")

# Inicializujeme proměnnou pro desítkové číslo
desitkove_cislo = 0

# Inicializujeme proměnnou pro pozici číslice
pozice = 0

# Projdeme každou číslici v binárním čísle od konce
for cislice in binarni_cislo[::-1]:
    # Převedeme číslici na celé číslo
    cislice = int(cislice)

    # Vypočítáme hodnotu číslice v desítkové soustavě a přičteme ji k výsledku
    desitkove_cislo += cislice * (2 ** pozice)

    # Zvýšíme pozici o 1
    pozice += 1

# Zobrazíme ekvivalentní desítkové číslo
print("Desítkové číslo je:", desitkove_cislo)
```

```
def je_prvocislo(cislo):
    """Funkce určí, zda je číslo prvočíslo."""
    if cislo <= 1:
        return False
    for i in range(2, int(cislo**0.5) + 1):
        if cislo % i == 0:
            return False
    return True


def hlavni_program():
    """Hlavní program, který přečte číslo od uživatele a
zobrazí, zda je prvočíslo."""
    cislo = int(input("Zadejte celé číslo: "))
    if je_prvocislo(cislo):
        print(f"Číslo {cislo} je prvočíslo.")
    else:
        print(f"Číslo {cislo} není prvočíslo.")


if __name__ == "__main__":
    hlavni_program()
```

```
def je_palindrom(s):
    """Rekurzivní funkce, která určí, zda je řetězec palindrom."""
    # Prázdný řetězec nebo řetězec s jedním znakem je palindrom
    if len(s) <= 1:
        return True
    # Pokud první a poslední znak nejsou stejné, není to palindrom
    if s[0] != s[-1]:
        return False
    # Rekurzivní volání pro zbytek řetězce bez prvního a posledního znaku
    return je_palindrom(s[1:-1])


def hlavni_program():
    """Hlavní program, který přečte řetězec od uživatele a určí, zda je palindrom."""
    retezec = input("Zadejte řetězec: ")
    if je_palindrom(retezec):
        print(f"Řetězec '{retezec}' je palindrom.")
    else:
        print(f"Řetězec '{retezec}' není palindrom.")


if __name__ == "__main__":
    hlavni_program()
```

```
def spravne_delitele(n):
    """Funkce, která vypočítá všechny správné dělitele čísla
n."""
    delitele = []
    for i in range(1, n):
        if n % i == 0:
            delitele.append(i)
    return delitele


def hlavni_program():
    """Hlavní program, který přečte číslo od uživatele a
zobrazí jeho správné dělitele."""
    cislo = int(input("Zadejte kladné celé číslo: "))
    delitele = spravne_delitele(cislo)
    print(f"Správné dělitele čísla {cislo} jsou: {delitele}")


if __name__ == "__main__":
    hlavni_program()
```