UNIVERZITA PALACKÉHO V OLOMOUCI

PEDAGOGICKÁ FAKULTA

Katedra technické a informační výchovy

# Bakalářská práce

Miroslav Juriček, DiS

# Sada úloh pro programování v jazyce Python na ZŠ

Olomouc 2023 vedoucí práce: doc. RNDr. Petr Šaloun, Ph.D.

**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 18. dubna 2023 …….

vlastnoruční podpis

**Poděkování**

Rád bych zde poděkoval vedoucímu bakalářské práce doc. RNDr. Petru Šalounovi, Ph.D. za jeho odborné rady a čas, který mi poskytl při řešení problémů vzniklých při psaní této bakalářské práce. Dále děkuji Mgr. Tomáši Dragonovi za poskytnutí cenných rad, jak psát kvalifikační práce.

Miroslav Juriček, DiS

# Obsah

# ÚVOD

Téma bakalářské práce vzniklo spojením několika klíčových faktorů, bez kterých bych si ho rozhodně nevybral. Při mém předchozím studiu jsem se programování v Pythonu podrobně věnoval. Tyto znalosti byly dále prohlubovány v rámci povinných i volitelných předmětů na univerzitě. To vyústilo v myšlenku, že právě toto téma by mohlo být vhodnou volbou pro mě. Společně s vedoucím této bakalářské práce jsme měli vizi konceptu, jehož očekáváným cílem nebyla další bakalářská práce na téma programování se všemi již zažitými stereotypy, nýbrž by měla vzniknout práce zabývající se odlišným pohledem na výuku programování. Vzhledem k tomu, že i učebnic s tématem programování v Pythonu je nepřeberné množství, má tato práce především ukázat jinou dimenzi výuky, než psát podle knížky staré a několikrát opakované koncepty jako například „Ahoj světe“.

Bakalářská práce je rozdělena do dvou částí. V teoretické části bylo cílem definovat možnosti vyučování programování v Pythonu na základní škole, analyzovat rámcový vzdělávací program a objasnit pojem zájmové vzdělávání. V rámci teoretické části je provedeno srovnání jazyka Python a Scratch, včetně jejich popisu. Poslední kapitolou tohoto celku je instalace programovacího jazyka Python, popis jeho vývojových prostředí, instalace knihoven. V rámci této kapitoly jsou porovnávána vývojová prostředí IDLE, PyCharm a Google Colaboratory.

Praktickou část tvoří dvě kapitoly. První se zabývá základními konstrukty Pythonu, což jsou například proměnné, seznamy, cykly nebo podmínky. Každá podkapitola obsahuje jeden příklad na procvičení daného celku, znění je uvedeno pod danou tématikou. Řešení je realizováno formou přílohy, ve které jsou vypracované a okomentované zdrojové kódy. V další kapitole jsou zapracovány úkoly k samostatnému řešení, některé vycházejí z učebnic pro jazyk Scratch.

Cílem této práce je ukázat možnosti, jak vyučovat programování v jazyce Python zajímavější formou. Kombinovat to, co žáci znají, např. matematické vzorce s novým učivem. Zbytečně žáky nepřehlcovat složitou syntaxí a realizovat pokud možno zájmové vzdělávání formou diskuse a ukázek, k čemu lze programování využít, jak může usnadnit práci a vzbudit větší zájem.

# 1 RVP a výuka programování v jazyce Python na ZŠ

Každá doba má nějaké specifické požadavky na vzdělávání. Podobně jako třeba ve středověku nebylo vyšší vzdělávání žádoucí, ale bylo potřeba rytířů, zbrojařů nebo platnéřů. Od objevení tranzistorového jevu a vynaleznutí tranzistoru se začalo rozvíjet vědní odvětví digitální technika, ze kterého se později vyčlenily informační technologie. Paralelně s tímto odvětvím šlo ruku v ruce programování a algoritmizace. To se postupně oddělilo v samostatnou vědní disciplínu nazývanou softwarové inženýrství. V dnešní době digitálních technologií je stále narůstající trend poptávky na trhu práce po programátorech a pracovnících v oblasti informačních technologií. Než vyšla revize rámcového vzdělávacího programu, zkráceně RVP, bylo náplní předmětů jako informatika, informační technologie nebo výpočetní technika především naučit žáka používat kancelářský balík MS Office, obsluhovat poštovního klienta, popsání jednotlivých částí počítače. Současnou revizí se školství snaží reagovat právě na zvýšenou poptávku po IT pracovnících. Ba co víc, většina společností má jako hlavní bod požadavků na uchazeče práci s PC. Pod tímto si můžeme představit prakticky cokoliv, a tudíž většinou záleží na konkrétních požadavcích na pracovní pozici. Podobně jako u lékaře je i práce programátora či IT specialisty celoživotním vzděláváním. U programátorů nastává jistá degradace mozkové kapacity, kdy je mnoho mladších programátorů vhodnějších pro pozici vývoje, neboť jejich myšlení neobsahuje například rutinní uvažování, které v této oblasti není žádoucí. Zkušenější programátoři zastávají především pozice vedoucích týmů či projektů. Proto je obrovská poptávka především po mladých programátorech, kterých je dlouhodobě nedostatek.

V novém RVP se snaží Ministerstvo mládeže a tělovýchovy (dále MŠMT) zohlednit tyto potřeby, a proto již od první třídy přichází žáci do kontaktu s nejrůznějšími programovatelnými hračkami, jako jsou například Bee-Bot nebo Ozobot. Postupně přecházejí na složitější úlohy, jako je třeba tvorba jednoduchých programů v graficky orientovaných jazycích, například Scratch. Souhrnně je tato oblast označována jako Informatika, která je členěna do 4 základních pilířů, které se jmenují Data, Informace a modelování, Algoritmizace a programování, Informační systémy, Digitální technologie [1]. Lze tudíž předpokládat, že absolvent základní školy bude nejen schopný ovládat počítač. Tyto kroky mají v žácích za úkol vzbudit zájem o programování, algoritmizaci a automatizaci. Dle mého názoru je toto poněkud nešťastné, neboť může dojít k budování nenávisti u žáků, kteří nejsou nadáni informatickým myšlením, avšak prozatím mi nepřísluší toto posuzovat a není to tématem této práce. Vzhledem k tématu této bakalářské práce je důležité v této kapitole zjistit, jakým způsobem a jestli vůbec

lze na ZŠ vyučovat programovací jazyk Python, aniž bychom tím neporušili pravidla či učební osnovy. Další otázkou je i budoucí využití umělé inteligence. Je totiž z pohledu zaměstnavatele zajímavé, že žáci budou umět programovat, avšak je zde zásadní problém, jestli by se nemělo zapracovat i na stránce formulování problémů a hypotéz. Už v dnešní době je umělá inteligence schopna psát programy, recept na vaření nebo popis výrobku. Pro správnou funkci, potřebuje exaktně definovaný uživatelský vstup. Vzhledem ke svým současným zkušenostem můžu říct, že i žáci na střední škole mají problém svůj dotaz správně formulovat.

Nyní se vrátím zpět k zásadnímu otázce, a to je, jak vyučovat programovací jazyk Python na základní škole. Jakožto pedagogičtí pracovníci máme spoustu práv, jako například volbu výukové metody, avšak RVP, potažmo ŠVP je pro nás závazné a nemůžeme si ho upravovat, jak se nám zlíbí. Proto je nemožné vyučovat programování v Pythonu v rámci standardní rozvrhové akce povinné školní docházky. Pokud bychom se přesto rozhodli tohle praktikovat, tak to do jisté míry může fungovat, ale v případě kontroly či stížnosti ze strany rodičů se jedná o problém, který budeme muset řešit a z právního hlediska se jedná o porušení pracovní kázně. Jediným možným východiskem je realizace této výuky formou zájmového vzdělávání. Pro něho totiž RVP není natolik závazné. Toto zájmové vzdělávání bývá častěji známo jako kroužek, například sportovní a pohybové hry nebo pěvecký sbor. Zájmové vzdělávání je ale exaktně definováno v dokumentech, které jsou k nahlédnutí na stránkách MŠMT.

## 1.1 Zájmové vzdělávání z pohledu zákona

Zájmové vzdělávání má podobně jako všechno, co souvisí s výukou, svou oporu v zákoně, přesněji se jím zabývá zákon číslo 74/2005 Sb., a ve své revizi zákon číslo 111/2022 Sb. a stanovuje, kdo je účastník zájmového vzdělávání, jaké jsou podmínky uskutečnění zájmového vzdělávání. Také uvádí druhy školských zařízení pro zájmové vzdělávání, mezi které se řadí například i školní družina, což pro mě osobně při tvorbě této bakalářské práce bylo velkým překvapením. Dále je zde zakotveno ustanovení o úplatě, požadavcích na vyučujícího a podobně. Celým zněním těchto zákonů bych se nechtěl více dopodrobna zaobírat. Vypíši zde pouze ty body, které plně souvisí s tématem této bakalářské práce.

V první řadě zákon přesně říká, jaký druh zájmové činnosti budeme realizovat. Pro nás budou závazné body *a) pravidelnou zájmovou, výchovnou, rekreační nebo vzdělávací činností včetně možnosti přípravy na vyučování* a *e) individuální práce zejména vytváření podmínek pro rozvoj nadání dětí, žáků a studentů.* Pro realizaci našeho „kroužku“ si musíme zvolit

formu, jakou se budeme prezentovat. Na výběr jsou dvě možností: středisko volného času nebo školní klub. Obě tyto formy mají své výhody, nevýhody a formální náležitosti. Osobně bych si zvolil klub, neboť na rozdíl od střediska je určen přednostně pro žáky druhého stupně základní školy a žáky nižších stupňů gymnázií. Podmínkou pro přijetí do klubu je dle zákona písemná přihláška. Organizační členění klubu stanoví vždy předpisem ředitel školy. Je tedy nutné ho o našem záměru informovat. Je důležité zmínit i to, že zákon jasně ukládá, jak často by se měla schůzka klubu organizovat. Z mé vlastní zkušenosti vím a sám bych se přikláněl k volbě rozsahu dvou hodin za týden [2].

## 1.2 Motivace žáků k návštěvě zájmového vzdělávání

Každý den se setkáváme s mnoha situacemi. Jsou takové, které nás dokáží motivovat i demotivovat. Například špatná známka dokáže vzbudit především negativní emoce, kdy v tento okamžik nepomýšlíme na žádnou motivaci. Stejně naši motivaci může ovlivnit rodinná situace nebo zdravotní potíže buďto nás samotných, popřípadě rodinného příslušníka. Nastává otázka, jakým způsobem namotivovat žáky a jak je přesvědčit, že právě výuka programování v rámci klubu je pro ně výhodná a přínosná. Přeci z pohledu žáka, který bude této činnosti věnovat čas po standardní vyučovací době a nebude kvůli tomu moci například trávit o hodinu více hraním her na počítači nebo venkovními aktivitami s kamarády, je tato nabídka naprosto neatraktivní. Jsem toho názoru, že jakožto pedagogové musíme být i dobrými marketéry. Když řekneme například na začátku hodiny „Žáci, musíte udělat program, se kterým půjdete na soutěž, jinak mi sem ani nechoďte“, tak to bude znít opravdu šíleně a pozitivní atmosféru tím nevzbudíme. Přesto jsou vyučující, kteří toto praktikují a prochází jim to. Což je pak ale na pováženou, jestli je zájmem žáků opravdový a jedná se o dobrovolnou činnost, nebo je tato „iniciativa“ spíše z donucení společnosti nebo rodičů.

Myslím si, že nejlepším způsobem, jak žáky přirozeně motivovat, je ukázka popřípadě přednáška, co všechno se můžou naučit nového, k čemu jim takové znalosti a dovednosti budou, jak je mohou využít při každodenním životě. Tím vzbudíme přirozenou zvídavost. Vyzdvihnout všechna pozitiva, jako například, že tento kroužek jim může ulehčit další působení v rámci střední školy nebo budou mít rozšířenější obzory více než ostatní žáci, kteří se kroužku neúčastní. Tento poslední jmenovaný argument je značně zavádějící, protože tím můžeme vzbudit pocit segregace mezi žáky z kroužku a ostatními. Je tedy potřeba volit vhodná slova a s motivací to nepřehánět, avšak zdůraznit, že tímto nedělíme skupinu na účastníky kroužku a neúčastníky. Koneckonců, každý den se setkáváme s reklamou, tak proč si z toho jako vyučující

něco neodnést. Máme možnost říct o tomto kroužku žákům, u kterých si všimneme, že mají jisté nadání či nadšení v dílčích hodinách. I tohle je způsob, jak některé jedince, kteří o tom doposud nepřemýšleli, oslovit a přitáhnout jejich pozornost. Je dobré používat i vhodné metody výuky, atraktivní témata pro žáky nebo ukázku praktických řešení. Je tedy namístě vypustit nadbytečnou teorii a zvážit moderní trendy, jako je projektová výuka, protože žáky především zaujmeme tím, co je zajímá než nezáživnými tématy [3].

## 1.3 Realizace zájmového vzdělávání zabývající se programováním

Při realizaci je třeba držet se podobných zásad jako při výuce jakéhokoliv jiného předmětu. Začít všeobecnými základy, vysvětlit základní konstrukce programovacího jazyka, jejich funkcionalitu včetně použití. V této úrovni je důležité naučit žáky správným programátorským návykům, syntaxi a porovnat výrazy s významem, který znají z jazyka Scratch. Určitě není vhodné chtít po žácích, aby se něco učili nazpaměť, neboť ani programátor s mnoha lety praxe není chodící encyklopedie. Tudíž je namístě připravit si vhodnou literaturu a baterii úloh, ze které budeme čerpat. Další otázkou je očekávaný výstup této činnosti. Vzhledem k faktu, že se jedná o zájmové vzdělávání s dobrovolnou účastí, by neměl být na žáky vyvíjen nátlak a všechno by mělo mít svůj přirozený vývoj. Je potřeba počítat s tím, že ve skupině budou žáci, kteří dokáží nové poznatky vstřebávat rychleji ale zároveň ti, kteří budou pomalejší.

Dovolil bych si v případě velkého rozptylu doporučit, aby tito nadanější a rychlejší žáci pomáhali s řešením vzniklých problémů těm pomalejším, neboť kooperativní učení je v dnešní době velký trend. Taktéž se tím prohloubí spolupráce a týmový duch a právě, když všechny komponenty do sebe zapadají tak, jak mají, můžeme i ze zájmového vzdělávání vytvořit funkční a velice efektivní celek, ve kterém pozitivní vztah mezi jednotlivými žáky bude důležitým faktorem a silným argumentem vzbuzujícím zájem ostatních o tuto výuku. Právě dobré pracovní klima je v dnešní době pro spoustu lidí rozhodující napříč profesemi, zájmovými útvary či pracovními buňkami. Existují výzkumy, které ukazují, že pro velké procento respondentů je důležitější do určité míry pracovní klima než odměna za vykonanou práci. Je třeba na to pohlížet podobnou optikou i ve školním prostředí a nastavit své cíle právě tak, aby bylo klima pozitivní a příjemné pro všechny účastníky. V opačném případě může nastat efekt odlivu zájemců, když budeme mít přehnané nároky a negativní klima v kolektivu. Tohle je zkrátka neodmyslitelný fakt, který funguje ve většině kolektivů, ať už se jedná o třídu, pracovní kolektiv nebo výzkumnou skupinu [3].

# 2 Dělení programovacích jazyků

Aby žáci plně pochopili, proč bude obsahem zájmového vzdělávání Python, je důležité řádně jim vysvětlit, proč se používá právě tento jazyk a jaké má výhody proti běžně vyučovanému graficky orientovanému jazyku. V této kapitole popíši rozdíl mezi strojově orientovaným jazykem a jazykem vyšší úrovně, obsahem následujících podkapitol je představení obou jazyků, včetně jejich vzájemného porovnání.

Podle míry abstrakce dělíme programovací jazyky na nízkoúrovňové a vysokoúrovňové. Mezi jazyky nízké úrovně patří Assembler neboli jazyk symbolických adres. Hlavním poznávacím znakem je především nutnost znalosti obsahu daného hardwaru, protože programátor má úplnou kontrolu nad tím, co se bude vykonávat, například součin dvou čísel a uložení tohoto výsledku do jiného registru. Vysokoúrovňové jazyky jako je například C#, Python nebo Java už nejsou tolik hardwarově vázané, tudíž nemusíme znát přesnou adresu buňky, s níž chceme pracovat. Jazyky vysoké úrovně jsou svým zápisem neboli syntaxí mnohem bližší lidské řeči, což umožnuje zapisovat složitější operace, a přesto neztrácíme přehled o tom, co by mělo být vykonáváno.

*Ukázka tvorby funkce v jazyce Assembler pro mikroprocesor ATMEL AT89C2051*

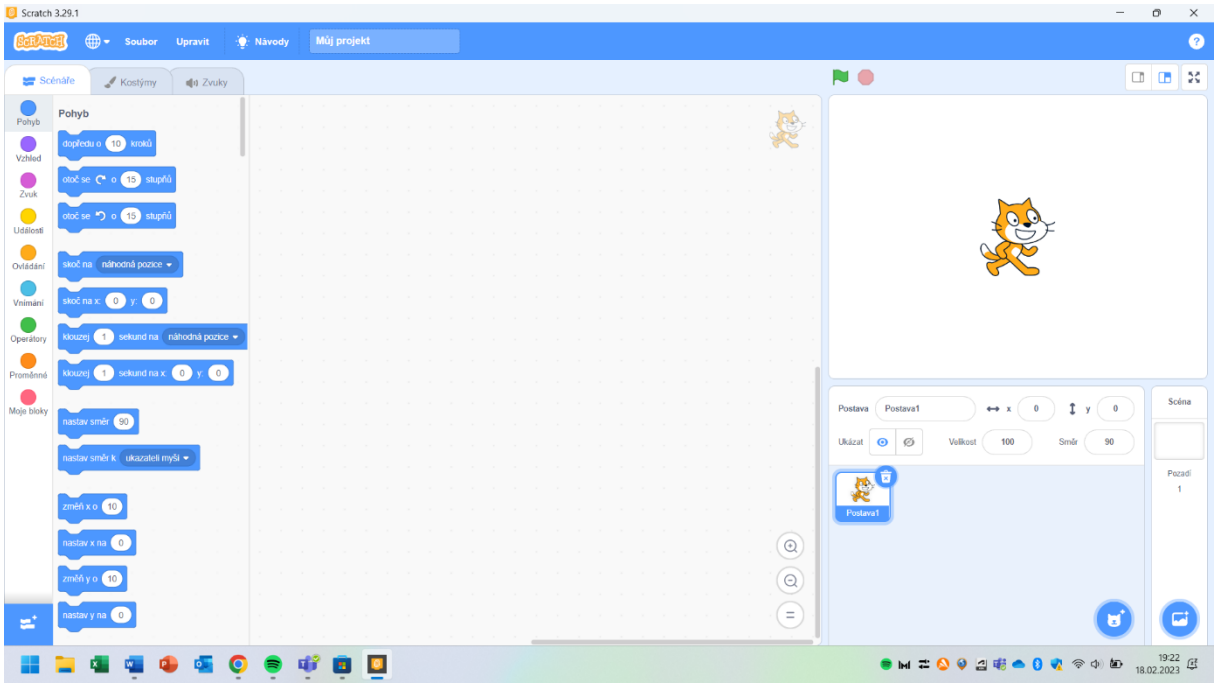
```
DEL1MS: push  PSW ;schovej puvodni RB
  orl PSW,#RB3 shl 3;nastav Register Bank = 3
  mov r7,#0
  djnz  r7,$
  mov r7,#0c6h
  djnz  r7,$
  pop PSW ;obnov Register Bank
  ret
```

(Zdroj: vlastní zpracování)

Programovací jazyky vysoké úrovně můžeme rozdělit na kompilované, například jazyk C, Java a interpretované, které zastupuje například Scheme. Toto dělení je dáno způsobem zpracování a vykonání samotného programu, kdy u interpretovaných jazyků jsou jednotlivé příkazy programu postupně překládány do strojových instrukcí a následně vykonávány. U kompilovaných je program pomocí kompilátoru převeden jako celek do spustitelného souboru. Oba tyto způsoby mají svá negativa i pozitiva, což vyústilo v použití kombinace interpretace a kompilace. Využívají toho například jazyky C# nebo Python [4].

## 2.1 Představení jazyka Scratch a jeho vývojového prostředí

Jazyk Scratch vznikl v roce 2005, jeho autorem je Mitchel Resnick a poslední verze 3.0 byla vydána roku 2019. Jedná se o vysokoúrovňový blokově orientovaný programovací jazyk. V rámci výuky informatiky v Česku je určen především pro druhé stupně základních škol. Jeho hlavní výhodou jsou tzv. bloky akcí, tudíž se uživatel neučí syntaxi, ale pouze do sebe logicky spojuje bloky v jeden funkční celek. Je dostupný ve více než 70 jazycích včetně češtiny a angličtiny. Scratch má své vývojové prostředí online na stránkách [scratch.mit.eu](scratch.mit.eu), nebo je dostupný v off-line verzi pro operační systémy Windows, Android, macOS a ChromeOS. Při otevření vývojového prostřední zde najdeme výchozí postavu kocoura jménem Scratch, který vykonává příkazy podle toho, co mu v blocích zadáme (naprogramujeme).




*Obrázek 1 Screenshot vývojového prostřední Jazyka Scratch*

(zdroj: [https://scratch.mit.edu/download](https://scratch.mit.edu/download), 2023)

V levé části je panel akcí nazvaný Scénáře, ve kterém najdeme jednotlivé bloky akcí. Pomocí těchto bloků můžeme s postavičkou kocoura pohybovat po ose x, y, provádět otáčení, dále je zde možnost přehrávání zvuků, nastavování parametrů vzhledu. Blok událostí je ekvivalentní pro objekty potažmo funkce v klasických programovacích jazycích. Následující blok ovládání je ekvivalentem pro podmíněné výrazy a cykly. Bloky vnímání jsou zaměřeny na reakci na nějakou akci (stisknutí šipky, kliknutí myší atd.). Další důležitou záložkou jsou operátory umožňující vkládání matematických operací, proměnné, které zprostředkovávají

jejich definici a následnou práci s nimi. Poslední záložkou bloků je tvorba vlastních bloků. V záložce kostýmy můžeme měnit výslednou podobu postavy, včetně vytvoření své vlastní s využitím fotek. Zvuků je nepřeberné množství a postava je může vykonávat tím, že jsou pomocí bloků naprogramovány. Vpravo od bloků se nachází pracovní pole, do kterého můžeme bloky vkládat a pracovat s nimi. Zelenou vlajkou úplně vpravo nahoře se spustí simulace programu, pomocí červeného osmiúhelníku simulaci zastavíme. Pod polem s kocourem najdeme správu postav, ve kterém můžeme měnit jejich parametry, vytvořit pozadí nebo přidat nové postavy do programu [5].

## 2.2 Představení jazyka Python

Python je vysokoúrovňový multiparadigmový open-source programovací jazyk. Aktuální verze je 3.11.1 a 2.7.18. Vyšel ve 3 různých verzích, které jsou vzájemně nekompatibilní – Python1, Python2 a Python3. Za vznikem stojí Guido van Rosum a je vyvíjen v Python Software Foundation. Podobně jako Scratch je i Python dostupný na nepřeberné množství platforem od Windows, přes macOS, Android až po Linux. Postupem času si získal velkou oblibu především díky své jednoduchosti zahrnující dynamickou kontrolu datových typů nebo velkému množství knihoven, které jsou tvořeny početnou komunitou profesionálních i hobby vývojářů. Lze ho kromě standartního programování využít pro tvorbu webových aplikací, tvorbu neuronových sítí nebo strojové učení. Pro běh programovacího jazyku Python ho nejdříve musíme nainstalovat do našeho zařízení a poté si zvolit vhodné vývojové prostředí. Toto je detailněji popsáno ve třetí kapitole. Výchozím formátem je textový soubor obsahující zdrojový kód a nesoucí příponu py. Základní konstrukty (klíčová slova) tohoto jazyka jsou psány v angličtině, tudíž je potřeba pro jeho výuku použít mezipředmětové vazby na znalosti technického anglického jazyka. Text kódu je odsazován, což může být především zpočátku složitější na pochopení [4].

Z vlastní zkušenosti jsem používal například knihovny `Tkinter`, což je základní knihovna Pythonu pro tvorbu grafického prostředí, dále jsem použil například `PyMySQL` nebo knihovnu `Turtle`. Práce s vyšším programovacím jazykem je zahrnuta v RVP pro střední vzdělávání, konkrétně v učivu *Práce s počítačem, operační systém, soubory, adresářová struktura, souhrnné cíle*. Během praxe na VOŠ a SPŠE v Olomouci jsem zjistil, že se žáci učí programovat v jazyce Python, především kvůli poměrně jednoduché, přehledné a snadno pochopitelné syntaxi, minimální potřebě řešení sémantiky, alespoň co se rozsahu výuky na střední škole týče. Dalším důvodem je celková vhodnost jazyka pro edukační účely, neboť

nevyžaduje znalost práce s hardwarem, řízení práce s pamětí. Je rozšířen napříč vývojářskou komunitou, což umožňuje hledaní vzniklých problémů na internetu [6].

*Ukázka programu na rozpoznání sudého a lichého čísla v jazyce Python*

```
cislo=int(input("Zadej číslo: "))
if (cislo %2):
  print("Je liché!")
else:
  print("Je sudé!")
```

(Zdroj: vlastní zpracování)

## 2.3 Porovnání programovacího jazyka Scratch a Python

Oba programovací jazyky jsou vysokoúrovňové, proto odpadá potřeba žákům vysvětlovat práci s pamětí a samotným hardwarem. Dalším společným znakem pro oba jazyky je uživatelská přívětivost, v podobě bloků v případě Scratche nebo poměrně jednoduchými konstrukcemi v jazyce Python.

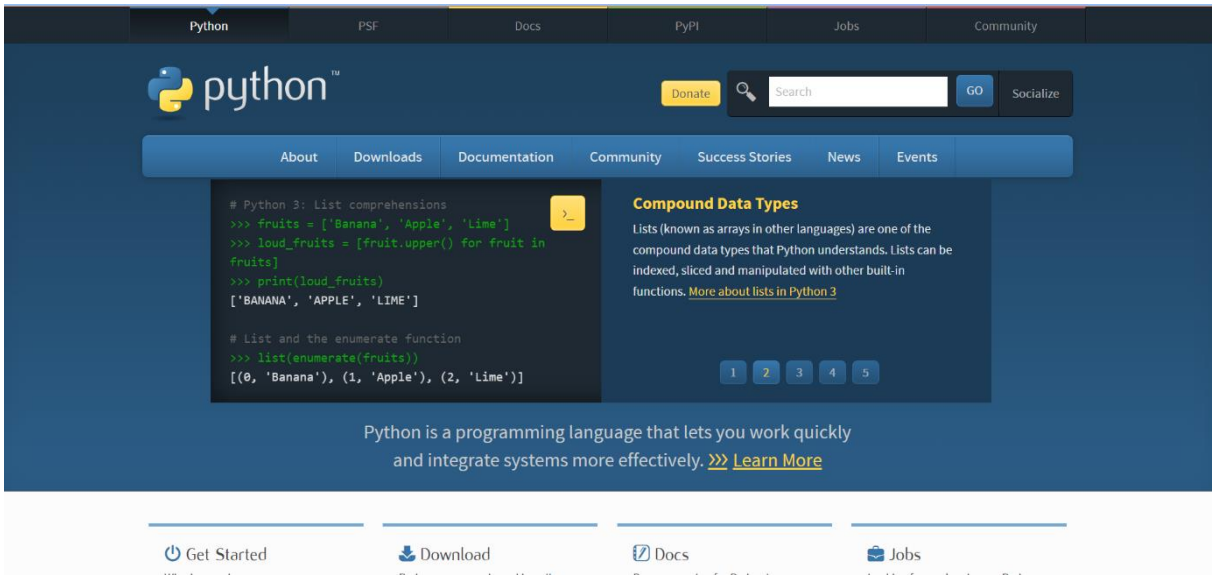
Díky své schopnosti poskytnout audiovizuální výstupy je Scratch vhodnější pro děti, protože multimediální prvky dokáží přitáhnout jejich pozornost a vzbudit zájem. Když žáci vidí, jak se jejich programy pohybují a vydávají zvuky, pozitivněji na to reagují. Vizuální pojetí společně s českým popisem bývá mnohem přijatelnější pro jejich dosavadní rozlišovací schopnosti, protože si snáze představí, co se skrývá pod názvy jednotlivých bloků. Nemusí se učit žádnou syntaxi a stačí jim pomocí vhodného tvaru do sebe spojovat jednotlivé bloky. Omezenost tohoto způsobu tvorby zdrojového kódu nahrává Pythonu. Je zde možnost tvořit si vlastní bloky, ale ty neumožňují dělat více než samotná nabídka po straně vývojového prostředí [3].

Python je především díky své uživatelské základně předurčen k tvorbě složitějších programů než Scratch. Má poměrně jednoduchou syntaxi, kterou lze dle mého názoru žákům 8. a 9. ročníků vysvětlit. V kroužku Micro:bitu, který aktuálně vedu, je několik žáků ve výše uvedených ročnících, kteří již přišli do kontaktu s Pythonem a jsou schopní vysvětlit s jistou přesností tyto základní konstrukce. Vysvětlení zápisu kódu může některé zájemce odradit a v tento okamžik je potřeba využít správnou motivaci. Nespornou výhodou Pythonu je jeho víceúčelovost, avšak není exaktně určen pro výuku programování. Další obrovskou výhodou je možnost spuštění programu na jiném operačním sytému bez nutností nejrůznějších exportů. Můžeme tvořit programy v různých paradigmatech a ukázat tak žákům rozdíl mezi procedurálním a objektově orientovaným programováním. Je třeba brát taktéž v potaz, že

defaultní výstup v Pythonu je textová konzole, což je další faktor, který je významným negativem ve srovnání s jazykem Scratch. Python nabízí množství hotových programů, které můžeme žákům ukázat, jedná se například o stahování videí z YouTube, vykreslování dat do mapy, práci s databázemi, práci s obrazem nebo jiná zajímavá témata, čímž můžeme pozitivně vzbudit přirozenou zvědavost a zájem. Takto náročné úkoly po žácích základní školy nemůžeme chtít. Pro práci jim můžeme dát rozpracovaný soubor, který společnými silami doplníme například o vzorce, nebo vstupní data. Takto můžeme některá negativa Pythonu překlopit do pozitiv a ukázat, že například „nudná“ konzole umí pracovat s multimédii.

# 3 Instalace programovacího jazyka Python

Programovací jazyk Python a jeho vývojové prostředí není běžnou součástí softwarového vybavení počítače. Pojem software obsahuje veškeré programové vybavení počítače, které můžeme rozdělit na systémový software a aplikační software. Pojem systémový software je program, který zajišťuje komunikaci mezi hardwarem a aplikačním softwarem, zahrnuje firmware, BIOS a operační systém. V této kapitole popíši instalaci pro operační systém Windows od společnosti Microsoft, se kterým se nejčastěji setkáváme ve školním prostředí. Instalace Pythonu je důležitá, pokud chceme v počítači spouštět aplikace psané v tomto jazyce. Pro vývoj těchto aplikací slouží vývojová prostředí (IDE), která jsou více rozepsaná v následujících podkapitolách [7].

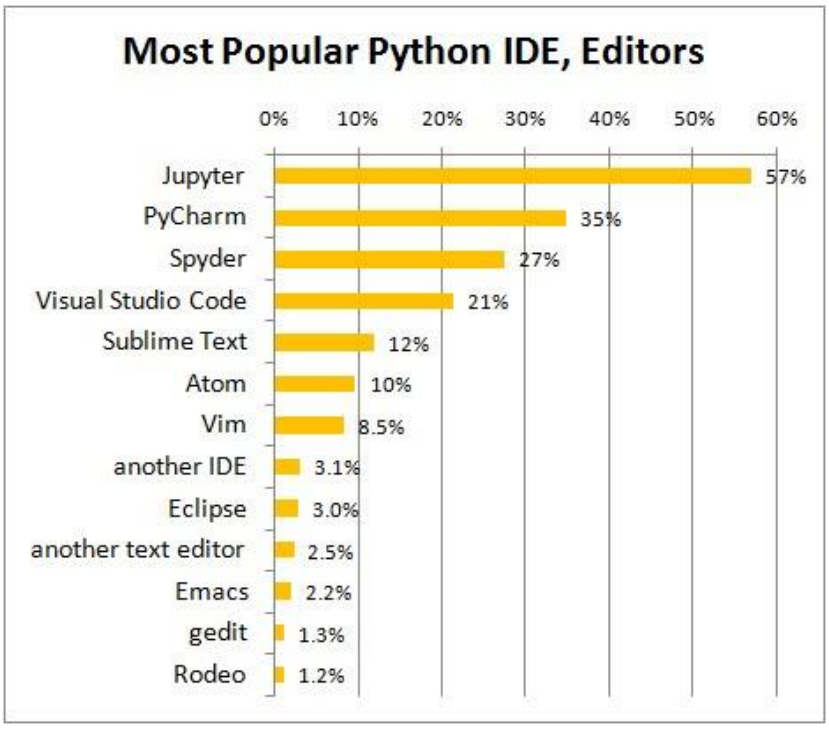



*Obrázek 2 Screenshot oficiální stránky programovacího jazyka Python*
(zdroj: https://www.python.org/, 2023)

Na oficiálních stránkách www.python.org najdeme základní menu, které můžeme vidět na obrázku 2. Pro tuto kapitolu bude nejdůležitější položka s názvem *Download*, pomocí které se zobrazí nepřeberné množství dostupných verzí Pythonu pro nejrůznější operační systémy od Windows, přes macOS po různé distribuce Linuxu. Pro akademické účely je rovněž zajímavá i položka *Documentation*, která je manuálem, jak programovat v Pythonu, ovšem tento manuál je psaný v angličtině, tudíž je již potřeba pokročilá znalost jazyka i v jeho technické podobě. Pokud je uživatel samouk, určitě by neměl minout položku *Community*, neboť obsahuje nespočet možností, jak navázat spolupráci s dalšími vývojáři. Po stisknutí položky *Downloads* > *Download for Windows* > *Python 3.11.2* se automaticky spustí stahování. Pozor, toto je

možnost, která stáhne soubor vhodný pro nejčastější uživatelskou konfiguraci, tzn. stolní počítač či notebook s klasickým procesorem a operačním systémem Windows 8.1 a novější. Pokud máme jinou verzi operačního systému nebo zařízení vybavené procesorem architektury ARM, musíme stisknout pouze položku *Downloads* a zde si vybrat verzi odpovídající našemu hardwarovému vybavení.

Jakmile se stáhne instalační soubor, je instalace již jednoduchá. Najdeme soubor v adresáři, do kterého jsme ho uložili a dvojitým poklepáním pravého tlačítka myši se spustí instalační aplikace, která extrahuje potřebné soubory do počítače, čímž provede instalaci. Pokud instalace proběhla bez chyby, která je nejčastěji způsobena souběhem více instalací najednou nebo omezenými právy uživatele, zobrazí se hláška o úspěšné instalaci. Obvykle se i s instalací programovacího jazyka Python nainstaluje jednoduché vývojové prostředí IDLE.




*Obrázek 3 Graf nejvyužívanějších vývojových prostředí pro Python*
(zdroj: https://morioh.com/p/a6236d3c9539, 2023)

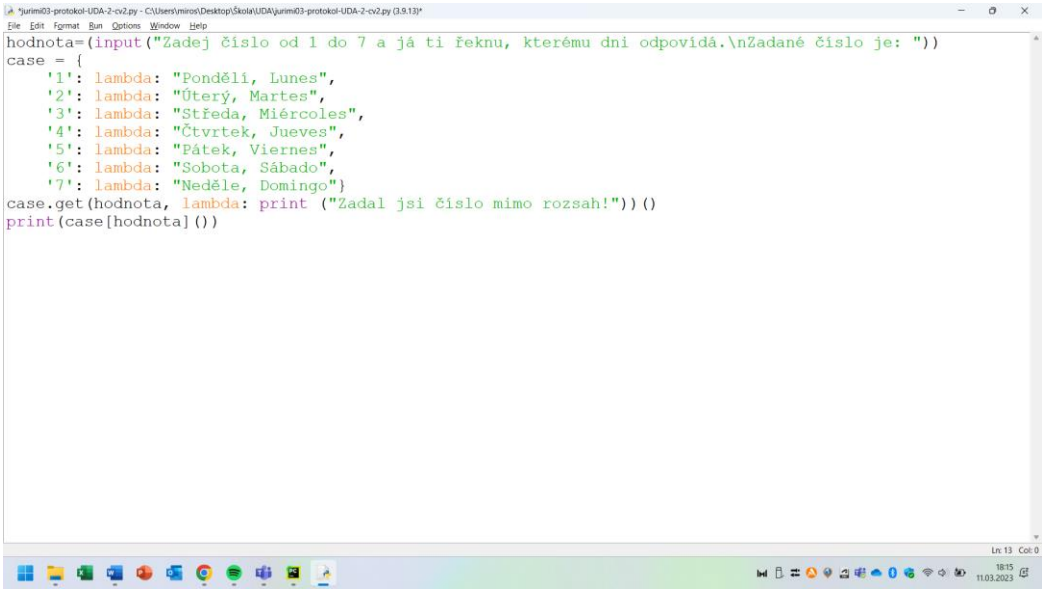
## 3.1 Vývojové prostředí Python IDLE

IDLE je nejjednodušší prostředí, určeno pro programování v jazyce Python. Obsahuje dvě části IDLE Shell, pro provádění interpretace kódu a Editor Window, ve kterém se píše a edituje kód. Aplikace samotná je naprogramovaná v Pythonu a pro grafické prostředí využívá knihovnu `Tkinter`.

IDLE nabízí především tyto funkce:

- psaní kódu v čistém Pythonu,
- prostředí je multiplatformní – funguje obdobně na Windows, Linux i macOS,
- okno IDLE Shell nabízí odbarvení vstupu, výstupu a chybových hlášení,
- textový editor stejně jako IDLE Shell nabízí odbarvení klíčových slov, textů a inteligentní odsazení,
- debugger s funkcí krokování pro odladění chyb.

Svým vzhledem IDLE připomíná textový editor. Obsahuje množství widgetů pro snazší tvorbu programů, jejich psaní a odlaďování. Nalezneme zde například možnost Debbugeru, v nápovědě manuály k programovacímu jazyku, nebo možnosti pro reset Shellu a spuštění napsaného kódu. Pravděpodobně nejdůležitější je widget *Option*, který umožňuje instalaci rozšíření, avšak možnost instalací knihoven je zde poměrně kostrbatá a uživatel již musí zvládat práci s PowerShell nebo příkazovou řádkou. Z tohoto pohledu je IDLE nešťastný, nabízí ale skvělý odrazový můstek pro procvičení základních konstruktů. Zajímavostí kromě manuálů ve widgetu *Help* je *Turtle Demo*, ve které najdeme ukázky želví grafiky, čímž žáky můžeme zaujmout. K nahlédnutí jsou vypracované ukázkové kódy a rovněž i možnosti vykreslení obrazců [8, 9].




*Obrázek 4 Screenshot vývojového prostředí IDLE s ukázkou vlastního kódu*

(zdroj: vlastní zpracování, 2023)

## 3.2 Vývojové prostředí PyCharm

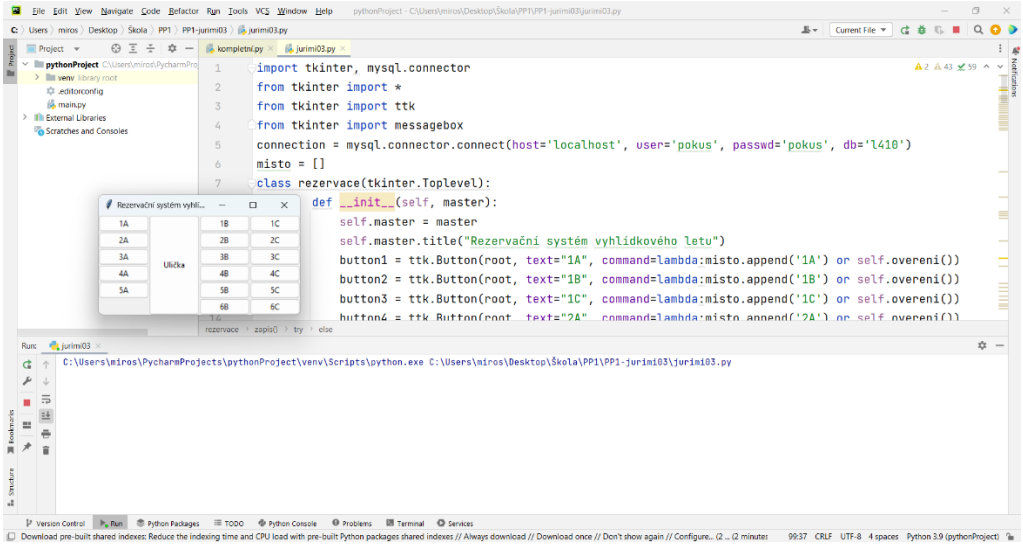
S prostředím PyCharm jsem se poprvé setkal až během prvních předmětů zaměřených na programování na univerzitě. Do té doby jsem používal výše zmíněný IDLE, nebo Visual Studio Code. PyCharm je Integrated Development Enviroment (IDE), které nabízí nespočet nástrojů pro tvorbu aplikací v programovacím jazyce Python, tvorbu webu a nabízí možnost programovat v jazyce Python 2 ve verzi 2.7, nebo Python 3 od verze 3.6. Pro uživatele je dostupný ve dvou verzích: Community a Professional. S prostředím Community přichází uživatel nejčastěji, jelikož se jedná o komplexní balíček, který je distribuován zdarma a je v open source verzi. Nabízí vše, co začínající programátor potřebuje včetně našeptávače kódu, debuggeru nebo možnosti instalace nejrůznějších knihoven. Ve verzi Professional lze vytvářet i webový obsah, podporuje blok Jupyter nebo nástroje pro programy, které jsou náročné na uložiště. Verze Professional podporuje HTML, JavaScript nebo tvorbu aplikací Flask nebo Django. IDE PyCharm není vázáno na konkrétní platformu, je tedy dostupné na Windows, MacOS nebo Linux [10].

Vzhledem k tomu, že sám aktivně používám verzi Community, se kterou budu i nadále pracovat, budu v následujících větách popisovat pouze tuto verzi a nikoliv Professional. Při prvním spuštění vývojového prostředí není vytvořen žádný projekt. IDE k tomuto kroku vyzve automatiky a poslední zavřený projekt pak při každém spuštění otevře. V dialogovém okně si nastavíme umístění projektu na disku, stejně jako si vybereme verzi jazyka Python, ve které budeme programovat a správnou verzi interpretu příkazů. Toto je důležitý prvek, protože špatná verze jazyka může později řádně komplikovat spuštění programu. Po založení projektu se spustí editor, který je rozdělen na tři základní části. Vlevo nahoře se nachází projektové okno, ve kterém jsou zobrazeny veškeré soubory projektu. Vpravo nahoře najdeme editor kódu, ve kterém podobně jako v IDLE, je kód barevně odznačen, odsazen a lze intuitivně poznat, co je příkaz, klíčové slovo, třída nebo text. Ve spodní části je konzole, tedy Shell. Konzole vypisuje chybové hlášky, vstupy a výstupy programu a podobně.

Podobně jako IDLE, tak i PyCharm pracuje ve výchozím režimu se základními knihovnami, jako jsou například `random`, `math`, `turtle` nebo `tkinter`. Pokud uživateli tyto základní knihovny nestačí, potřebuje projekt rozšířit a pracovat například s knihovnou PyMySQL, musí ji ručně doinstalovat. Existují na to dvě možnosti. Složitější a méně používaná je metoda instalace přes příkazovou řádku. Uživatelsky přijatelnější je přes dialogové okno, do kterého se dostaneme následovně: *File* > *Settings*. V tomto nastavení si najdeme položku *Project: pythonProject*, ve které je podpoložka *Python Interpreter*. Zde klikneme na malé plus,

které otevře dialogové okno, které je rozdělené na tři části. Navrchu je ikona lupy, reprezentující vyhledávač, do kterého zadáme jméno knihovny, nebo funkci, kterou hledáme. V levé části je rolovací menu obsahující seznam nalezených knihoven, v pravé pak popis, číslo verze a rok vydání. Pomocí tlačítka *Install Package* se knihovna nainstalujeme do projektu. Tato instalace se nemusí provádět při každém otevření projektu, pro který byla instalována.

Podobně jako v IDLE, je i zde navigační menu pro práci s kódem, souborem atd. Nejdůležitější je widget *File* pro správu souborů a možnosti exportu nebo tisku. Dalším důležitým prvkem pro spuštění a ladění programů je widget *Run*, který obsahuje možnosti spouštění programu nebo ladění, tedy Debug. Velice návykové je aktivní používání klávesových zkratek, které umožní omezit pohyby myší a zvýší míru soustředění na psaní programu. Vedle výše zmíněnému, je zde widget *VCS*, který umožňuje ihned publikovat na sítích jako je GitHub nebo Space. Ostatní widgety nemá podle mého názoru smysl více popisovat, neboť jsou velice podobné právě prvně zmíněnému IDLE.




*Obrázek 5 Screenshot vývojového prostředí PyCharm s ukázkou vlastního kódu rezervačního systému*
(zdroj: vlastní zpracování, 2023)

## 3.3 Google Colaboratory

Google patří podobně jako Microsoft k velikánům na poli vývoje softwaru a udělal velký krok vpřed, co se týče bezplatných cloudových řešení a balíků kancelářských aplikací. Aplikace Google Colaboratory mnou byla dlouho dobu od svého zavedení zavrhována, protože nedosahovala ani kvalit vývojového prostředí IDLE. Během výuky předmětu Didaktika programování jsem s ní přišel znovu do kontaktu. Pozitivním zjištěním bylo, jak se z původně jednoduchého vývojového prostředí stal vyzrálý a téměř plnohodnotný nástroj.

Google Colaboratory, zkráceně Colab, je bezplatný a volně dostupný nástroj pro psaní programů a skriptů v Pythonu. Oproti předchozím zmíněným vývojovým prostředím nevyžaduje pro svou činnost instalaci, neboť se jedná o prostředí dostupné online v cloudovém režimu, takže i program vykonává virtuální stroj. Předlohou pro vývojové prostředí je Jupyter Notebook, který patří k nejpopulárnějšímu vývojovému prostřední. Obrovskou a nespornou výhodou je plná kompatibilita a napojení na ekosystém od společnosti Google, díky čemuž umožňuje sdílení souborů napříč pracovní skupinou. Musíme toto prostředí tedy brát jako mezikrok mezi světem graficky orientovaného programování a tvorbou programů v komplexních vývojových prostředích. Colab podporuje jazyk Python 3 ale i starší verzi Python 2. Jediným omezením této služby je nutnost registrace do ekosystému Googlu, což pro většinu uživatelů mobilních telefonů s operačním systémem Android nepředstavuje překážku. Dalším velkým pozitivem je možnost v jednom dokumentu psát spustitelný kód, k němu následně psát text s obrázky nebo obsahem HTML. Toto je zajímavá alternativa k LaTeXu, pokud by autor uvažoval o tvorbě publikace o programování.

Hlavní nevýhodou je nutnost přístupu k internetu a omezenost platformy, co se rozšiřitelnosti o knihovny týče. Taktéž se mi úplně nelíbí strojové učení, neboť na základě mého myšlení může Google nabízet mé myšlenkové pochody třetí osobě.

Uživatelské prostředí je podobné jako v IDLE. Ve vrchní části je panel nástrojů pro práci se souborem, úpravy, spuštění programu. Po levé straně je seznam proměnných, práce s adresářem a podsoubory, popřípadě hledat. Ve spodní části je lokalizována vstupně/výstupní konzole. Uprostřed okna se nachází blok, do kterého programátor zapisuje program. Pomocí tlačítka „+“ je možné přidat buďto další spustitelný kód nebo textový obsah.


*Obrázek 6 Ukázka vlastního programu ve vývojovém prostředí Google Colaboratory*

(zdroj: vlastní zpracování)

## 3.4 Výběr vhodného prostředí pro zájmové vzdělávání

Do zájmového vzdělávání se obvykle hlásí žáci, kteří mají o danou problematiku zájem, je potřeba se při výběru vhodného vývojového prostředí držet několika důležitých aspektů. Nejdůležitějším bude vybavení učebny, ve které bude výuka realizována. Je-li k dispozici učebna s modernějším vybavením, budeme se zabývat jen stránkou softwaru. Může se ale stát, že učebna bude nedostupná nebo může obsahovat nedostatečné vybavení pro instalaci vývojového prostředí a spuštění aplikací. V tomto případě bych doporučil Google Colab, který poslouží dostatečně avšak je třeba brát na zřetel, že nemusíme naplnit všechny očekávané výstupy, které jsou v praktické části této bakalářské práce.

V případě, že máme k dispozici vybavenou učebnu s modernější technikou doporučil bych vývojové prostředí PyCharm z několika důvodů:

- především v tomto prostředí máme prakticky neomezený přístup k nejrůznějším knihovnám, což nám rozšiřuje programátorskou abstrakci,
- propracovaný systém ladění programu, který je z mé osobní zkušenosti nejlepší ze všech tří prostředí,
- doporučil bych vytvořit účet GitHub, kam bych jakožto vyučující publikoval své hotové kódy, což poslouží jako edukační materiál a případná kontrola nebo návod na správné řešení pro žáky (zde využijeme integrovanou funkci PyCharm,

publikování na GitHub, takže nemusíme složitě soubor nahrávat přes webový prohlížeč na cloud),

- anglická lokace, díky niž si žáci procvičí angličtinu, čímž využijeme mezipředmětové vazby,
- vývojové prostředí PyCharm se opírá o početnou programátorskou a vývojářskou základnu, která má narůstající trend, aktuálně je na 4. příčce dle webu pypl.github.io/IDE.html,
- jedná se o profesionální a komplexní nástroj, který uživatel využije v praxi, tudíž budou žáci připravení na budoucí práci programátora a osvojí si základy obsluhy tohoto prostředí, které je svým designem podobné Visual Studiu od společnosti Microsoft.

Při sumarizaci všech těchto argumentů dává vývojové prostředí PyCharm nejrozumnější balíček, který je na bezplatném trhu dostupný. Jedná se o nástroj, který je inteligentní, neboť barevně odděluje text od příkazů, automaticky odsazuje kód a zároveň již během psaní kódu upozorňuje na potencionální chyby. Přínosný může být rovněž i našeptávač syntaxe, jelikož lze u této zájmové skupiny předpokládat, že nebude mít syntaxi natolik osvojenou, aby ji psala bez hledání, čímž našeptávač ušetří čas. Rovněž IDE PyCharm patří mezi nejpopulárnější vývojová prostředí jak je možno vidět na obrázku číslo 3, takže i z tohoto titulu se žáci budou učit na tom „nejlepším“ co aktuálně trh nabízí [12].

# 4 Základní konstrukce jazyka Python

V předchozích kapitolách bylo shrnuto, jaké vybrat vývojové prostředí, co je vlastně jazyk Python. V této kapitole a následných podkapitolách jsou pospány základní konstrukce jazyka. Tato kapitola je zde zasazena z důvodu potřeby znalostí základní syntaxe jazyka Python, neboť bez ní nebudou žáci vůbec vědět, co daný kód vykonává nebo jak jej zapsat. S ohledem na omezené znalosti programovacího jazyka žáky, nemusí být popis konstrukcí naprosto správný a přesný. Uvedené formulace jsou psány s ohledem na pochopení pro výuku na základní škole a můžou být často zjednodušené s cílem zvýšit porozumění při výkladu. V návaznosti na procvičené konstrukce je vhodné zmínit, že každé cvičení může být realizováno i v jazyce Scratch. Vzhledem k obsahu kroužku, bych se nebál použít metodu experimentu, ve kterém by si žák vyzkoušel cvičení konstrukce v blocích a poté napsat v Pythonu. Formou diskuze může reflektovat své poznatky, popsat klady a negativa.

## 4.1 Proměnné

Python je typově dynamický programovací jazyk, což skvěle poslouží především v začátcích programování, protože se začínající programátoři nemusejí trápit s deklarací typu proměnné. To má své omezení, se kterými přicházíme do kontaktu při tvorbě složitějších programů, neboť některé funkce umí pracovat pouze s přesným datovým typem.

Proměnná je vyčleněné místo v paměti, které můžeme rozdělit na globální a lokální. Pro práci s každou proměnnou je třeba dbát pravidel, která můžeme najít v publikacích o programování, kterou jsou mimo jiné i Základy programování v Pythonu od pana doktora Martina Trnečky. Každá proměnná má své unikátní jméno, které je označováno jako identifikátor proměnné, právě pomocí tohoto jména můžeme s proměnnou pracovat v rámci rozsahu její platnosti. Jméno proměnné je obvykle odvozeno od toho, co má hodnota uvnitř reprezentovat. Jméno může být složeno z písmen, číslic a znaků. Pozor, existují ale rezervovaná slova, která nesmí být pro jméno proměnné použita.

**Definice hodnoty proměnné:**

```
a = "Toto je naše proměnná"
```

*Tabulka 1 Rezervovaná slova*

| | | | | |
|---|---|---|---|---|
| and | del | from | not | while |
| as | elif | global | or | with |
| assert | else | if | pass | yield |
| break | except | import | print | |
| class | exec | in | raise | |
| continue | finally | is | return | |
| def | for | lambda | try | |

(Zdroj: http://trnecka.inf.upol.cz/teaching/skripta-zpp1.pdf)

V tabulce 1 můžeme vidět, že se jedná především o slova, která jsou vyhrazena pro základní konstrukty, například cykly, výpis na obrazovku, import knihovny nebo výjimky. Pro přiřazení do proměnné použijeme logický výraz rovná se „=“, podobně jako je v ukázce definice proměnné. Do proměnné přiřazujeme zpravidla číselné výrazy, řetězce, seznamy atd. [4, 13]

## 4.2 Input, print

Každá z těchto funkcí má jiný úkol, avšak obě zajišťují komunikaci s uživatelem. Aby mohl program vykonávat svoje „tělo“, potřebuje zpravidla vstupní data, poté proběhne program a výstupem jsou opět data. Vstupem může být událost (čtení z klávesnice uživatele, kliknutí myší), CSV soubor nebo odkaz na video umístěné na YouTube. Výstupem může být okno s obsahem (např. textový výpis na obrazovce uživatele) nebo nový soubor. Prozatím se omezím na použití konzole a práci v ní. K tomuto účelu jsou vyhrazena klíčová slova `print` a `input`.

Funkce `print` je standartním výstupem jazyka Python. Společně s funkcí `return` se s nimi setkáme při tvorbě programů, protože výstupem programu je ve většině případů návratová hodnota nebo výpis. Nechť je příkladem vytvoření rezervace v kině. Objeví se dialogové okno, že byla vytvořena rezervace (`print`) a zároveň `return`, který vrací vybraná místa hlavnímu programu pro další zpracování v databázi a sdělení v emailu. Je zajímavé, že velká část publikací na téma programovaní v Pythonu obsahuje zmíněnou kombinaci `print` a `return`. Z hlediska didaktického to není zásadní poznatek ale z pohledu programátora ano. Ve značné části publikací je funkce `print` výstupem funkce. To je ale z programátorského hlediska nesprávné řešení. Pro výstup funkce a ve složitějších případech knihovny je jediná návratová metoda `return`.

**Příklad funkce print:**

```
print ("Ahoj světe!")
```

Zápis funkce `print` je jednoduchý, má svá úskalí, která je třeba vysvětlit a zmínit. Příkaz je složen z názvu funkce `print` a argumentu funkce *Ahoj světe!* Jako argument funkce ale nemusí být pouze řetězec, může to být hodnota nebo jejich kombinace.

**Ukázka výpisu textu a obsahu proměnné:**

```
text = "Eliška"
print ("Ahoj, tady", text)
```

Výsledkem toho příkazu bude vypsání textu: *Ahoj, tady Eliška.* Je nutno podotknout, že se jednotlivé argumenty oddělují mezerou a konec výstupu je zakončen tvrdým zalomením textu.

**Ukázka formátovacího řetězce**

```
text = "dědoušek"
jablka = 4
print(f"Měl {text} {jablka} jablka.")
```

Při využití formátovacího řetězce je umožněno do funkce `print` zkombinovat větší množství textu i proměnných. Důležité je vložit na první pozici v závorkách znak `f`. Výhoda tohoto řešení spočívá v možnosti úpravy oddělení textu, kdy není mezera přidána automaticky ale její vložení je plně závislé na zápisu programátora. Název proměnné se zapisuje do složených závorek. Pro podrobnější studium odkáži na literaturu.

**Příklad číslo 1:**

Doplňte do vytvořeného programu na okomentované řádky jména členů Vaší rodiny. Poté spusťte program a sledujte, co se bude vykonávat. Program s komentáři je součástí přílohy č.1, soubor s názvem *priklad_1-print.py*.

`Input` je vstupní funkce jazyka Python, charakteristická je uložením vstupních dat do předem definované proměnné, ve které jsou uložena a připravena k dalšímu zpracování.

**Příklad použití funkce input:**

```
rok = input("Zadej rok svého narození: ")
```

Výsledkem této funkce bude vložení hodnoty do proměnné rok. Funkce je složena z názvu *input* a argumentu funkce, v tomto případě textu *„Zadej rok svého narození:“*. Argument není povinný, program se spustí i bez něho. Pro přehlednost je doporučeno ho psát, protože může

nastat situace, kdy bude program čekat na vstup, a přitom se bude zdát, že není prováděna žádná instrukce.

**Příklad číslo 2:**

Do předpřipraveného programu doplňte vzorce pro výpočet povrchu a objemu válce. Použijte k tomu znalosti výšky a poloměru podstavy. Poté program otestujte na již vypracovaných úkolech z matematiky. V programu je nastaveno zaokrouhlení na dvě desetinná místa. Změnou hodnoty čísla u funkce `round` můžete zvýšit přesnost pomocí argumentu funkce, protože číslo udává na kolik desetinných míst bude výsledek zaokrouhlen. Program s komentáři je součástí přílohy č.1, soubor s názvem *priklad_2-print,input,vzorce.py* [4, 15].

## 4.3 Podmínky

Abychom mohli dělit vstupy nebo vykonávat operace podle parametru, budeme potřebovat větvení programu, se kterým se často setkáváme jako s podmínkami nebo podmíněným vykonáváním programu. Tělo instrukce podmínky je složeno ze 3 částí, první část je klíčové slovo `if`, druhou je podmínka, která je povětšinou interpretována logickým výrokem a třetí je blok instrukcí. Logický výraz se vyhodnocuje na dva stavy – `True`, pak říkáme, že podmínka byla splněna a dojde k provedení bloků instrukcí, nebo `False`, tedy podmínka nebyla splněna a blok instrukcí zůstává nevykonán. U podmínek je nutné dodržet správné odsazení, o které se při správném zápisu postará vývojové prostředí. Může nastat chyba, kdy zapomene napsat za logický výraz znak dvojtečky, v takovém případě se odsazování bloku instrukcí neprovede a program nepůjde spustit, protože nemá řádnou formu. Důrazně tedy doporučuji při výuce tuto častou chybu zmínit. Odsazování je doména Pythonu a je nutné tvořit již od začátku správné návyky. Špatné návyky se později hůře odstraňují.

**Příklad neúplného větvení:**

```
if podmínka:
  #blok instrukcí
```

Pokud použijeme pouze blok s `if`, jedná se o tzv. neúplnou podmínku. V případě, že chceme úplnou podmínku, přidá se do kódu větev s klíčovým slovem `else`. Na rozdíl od `if` neobsahuje podmínku. Obsahuje blok instrukcí a může být v daném větvení pouze jednou. Tento blok instrukcí se vykoná vždy, když se neprovede blok instrukcí za podmínkou `if` [4, 14, 16, 17].

**Ukázka úplného větvení:**

```
if podmínka:
  #blok instrukcí
else:
  #blok instrukcí
```

**Příklad číslo 3:**

Napište jednoduchý program, který určí, zda je uživatelem zadané číslo sudé nebo liché. Použijte všechny již probrané konstrukty, tedy podmínky, vstup a výpis. Sudé číslo je tehdy, pokud zbytek po celočíselném dělení (modulo) dvěma je roven nule. V pythonu je operace modulo interpretována znakem %. Program s komentáři je součástí přílohy č.1, soubor s názvem *priklad_3-podminky.py*.

Programátorovi nemusí nestačí ani dvě větvě programu. K tomuto slouží přepínač, realizovaný nepovinným příkazem `elif`. Tento komplexní nástroj konstrukcí `if-elif-else` najdeme v literatuře pod pojmem zřetězené podmínky. V podstatě si to můžeme představit jako třídičku na mince, kdy v logickém výrazu každé z podmínek bude velikost mince a v bloku příkazů, aby se její hodnota přičetla k celkové částce v pokladničce. Interpretace zřetězených podmínek probíhá tak, že interpret ověřuje všechny podmínky a až narazí na takovou, která je splněna, provede se její blok instrukcí. Podmínky do sebe můžeme libovolně vnořovat a vytvářet tím složitější konstrukce [4, 16, 17].

**Ukázka zápisu zřetězených větvení:**

```
if podmínka1:
  #blok instrukcí
elif podmínka2:
  #blok instrukcí
elif podmínka3:
  #blok instrukcí
else:
  #blok instrukcí
```

**Příklad číslo 4:**

S použitím matematických vzorců vytvořte program, který bude počítat kořeny kvadratických rovnic. Výsledek diskriminantu bude pomocí podmínek rozdělen na tři scénáře – rovnice nemá řešení v oboru reálných čísel, rovnice má jeden reálný kořen a rovnice má dva reálné kořeny. Pokud rovnice má řešení, vypíše program kořen/y. Kořeny zaokrouhlete na tři desetinná místa.

Pro druhou odmocninu využijte funkce z matematické knihovny `math.sqrt(číslo)`. Program s komentáři je součástí přílohy č.1, soubor s názvem *priklad_4-zretezene_podminky.py*.

**Příklad číslo 5:**

Napište program, který rozhodne, zda je uživatelem vložený rok přestupný nebo ne. Je-li číslo roku dělitelné číslem 4 beze zbytku a zároveň není beze zbytku dělitelné 100, pak se jedná o přestupný rok. Pokud je číslo roku beze zbytku dělitelné 100, pak se nejedná o přestupný rok. Pokud je číslo roku dělitelné beze zbytku 400, pak je rok přestupný. Ve všech ostatních případech se jedná o nepřestupný rok [4]. Program s komentáři je součástí přílohy č.1, soubor s názvem *priklad_5-podminky,logicke_vyrazy.py*.

## 4.4 Výjimky

S touto kapitolou obvykle nepřichází do kontaktu žáci základních ani středních škol, a přitom pomůže odstranit velkou část chyb v pokročilejších programech. Python je moderní jazyk, který obsahuje nejrůznější ochranné prvky, jejichž cílem je co nejvíce eliminovat chybu programátora. Rozeznáváme základní tři druhy chyb: sémantické chyby, ty můžeme pomocí výjimek odstranit, syntaktické, v případě, že je kód špatně napsaný nebo běhové, tzn. při vyhodnocování kódu dojde k chybě. Výjimky pomáhají chyby běhové a sémantické odstranit. Pokud se chcete dozvědět o této problematice více, doporučím literaturu například od pana Marka Lutze – Learning Python. Vzhledem k možnostem této bakalářské práce uvedu a vysvětlím pouze základní princip výjimek. Chybný vstup může být eliminován například pomocí podmínek `if`. Toto řešení není zcela správné a může nastat situace, že i přes několik podmínek dostaneme špatný uživatelský vstup. Abychom tomu předešli, použijeme klíčová slova `try` a `except`. Například jednoduchý kód, který bude dělit dvě čísla, kdy jako dělitele zadáme nulu. V takovém případě program skončí chybovou hláškou *`ZeroDivisionError: division by zero`* a právě tehdy je pro nás výhodné použití výjimek.

**Příklad zápisu výjimky:**

```
#tělo programu
try:
  #Pokusí se vykonat nějakou funkci nebo část kódu.
#Tato sekce je v literatuře označovaná jako hlídaná sekce
except:
  #V případě, že pokus skoční chybou, přidáme do této větvě např. chybo
vou hlášku.
```

Větví `except` můžeme mít volitelné množství a trochu připomínají podmínky. Můžeme si například nadefinovat výjimku pro každou chybovou hlášku tak, abychom věděli, jaká chyba nastala. Konkrétně tak, že za klíčové slovo `except` přidáme chybovou hlášku, například `except ZeroDivisionError` [18, 19].

**Příklad číslo 6:**

Napište program, který vypočítá kořen lineární rovnice (ax + b = 0). Vstupem programu budou dvě proměnné `a` a `b`. Použijte výjimky, neboť může nastat stav, kdy uživatel vloží oba vstupy nulové. V takovém případě bude ve větvi `except` napsáno, že uživatel vložil špatné parametry. Program s komentáři je součástí přílohy č.1, soubor s názvem *priklad_6-vyjimky*.

## 4.5 Cyklus while

Cyklus je příkaz, který umožňuje opakovat určitou část kódu. Cyklus `while` je složený ze dvou částí, podmínky a bloku příkazů. Jestliže je podmínka vyhodnocena jako pravdivá, vykoná se blok příkazů. Po vykonání je znovu vyhodnocena podmínka a dochází opět k ověření platnosti podmínky. V případě vyhodnocení podmínky jako pravdivé je proveden blok instrukcí. Pokud je podmínka vyhodnocena jako nepravdivá, blok příkazů se nevykoná a vyhodnocují se další výrazy, které jsou za cyklem. Každý průchod cyklem se nazývá iterace, která je číslována od nuly až po n. Cyklus `while` je ze své podstaty výhodný pro aplikace, u kterých neznáme přesný počet opakování [4].

**Ukázka syntaxe cyklu while:**

```
while podmínka:
  #blok instrukcí
```

**Příklad číslo 7:**

Vytvořte program, ve kterém bude uživatel hádat neznámé, náhodné číslo. Na začátku programu se vygeneruje náhodné číslo v uživatelsky definovaném rozsahu. Následně se bude cyklicky opakovat porovnání uživatelovy volby a náhodného čísla. Dokud uživatel neuhodne toto číslo, bude probíhat tělo cyklu, ve kterém bude pomocí podmínek probíhat porovnání náhodného a uživatelem zadaného čísla a bude mu poskytnuta nápověda, zda má zvolit další číslo nižší nebo vyšší. Program s komentáři je součástí přílohy č.1, soubor s názvem *priklad_7-cyklus_while.py*.

## 4.6 Cyklus for

`For` je v Pythonu cyklus, který má předem známý počet opakování. Obvykle je spojen s funkcí `range`, která vrací sekvenci celých čísel. `Range` je složen ze tří částí – počáteční hodnota, konečná hodnota a krok, který je ve výchozím stavu zvětšován (inkrementován) a jeho parametr není nutný. Pokud chceme, aby byl krok větší, například vykreslování desky pro šachy, může programátor použít i jiné než 1 nebo může použít i záporné číslo. Iterace probíhá tak dlouho, dokud není konečná hodnota rovna počtu iterací. Rovněž můžeme pro `range` použít číselnou hodnotu, pokud jde o přesný počet iterací. Python počítá neboli indexuje iterace od nuly do koncové hodnoty, ovšem koncovou hodnotu nepočítá. Hodnotu indexu si můžeme představit jako vzdálenost od počátku po konečnou hodnotu. Pokud má být zahrnuta i koncová hodnota, musíme ji zvětšit o 1, tj. použít výraz hodnota+1 [4, 18].

**Ukázka syntaxe cyklu for s použitím funkce range:**

```
for i in range (počáteční hodnota, koncová hodnota, krok):
  #blok příkazů
```

V praxi se můžeme setkat s funkcemi `len` a `enumerate`. Řetězec je objekt jehož základní vlastností je znalost skutečné délky řetězce (instance). Tohoto využívá funkce `len`. Funkce `enumerate` společně s použitím cyklu `for`, v každé iteraci cyklu vrací hodnotu indexu prvku v poli a zároveň hodnotu skrývající se pod tímto indexem do předem definovaných proměnných, které jsou součástí těla cyklu. Mějme seznam = [0, 12, 2, 3]. Pro index nula bude hodnota nula, pro jedna dvanáct atd. Cykly `while` a `for` lze do sebe libovolně vnořovat podobně jako v blocích. Do těla cyklů mohou být vnořeny i podmínky [8].

**Příklad číslo 8:**

Vytvořte program, který pomocí parametrů funkce `range` a cyklu bude vypisovat násobilku daného čísla. Uživatel bude vkládat pouze první číslo, jehož násobky se budou počítat. Dále vloží hodnotu maximálního násobku, do kterého se bude násobilka počítat. Krok nevyplňujte. Myslete na indexování a zahrnutí správné koncové hodnoty funkce `range`.
Program s komentáři je součástí přílohy č.1, soubor s názvem *priklad_8-cyklus_for.py*.

**Příklad číslo 9:**

Vytvořte program, jehož vstupem bude slovo nebo řetězec. Poté deklarujte proměnné samohlásky a souhlásky, které budou mít výchozí hodnotu 0. Pomocí cyklu projděte celé slovo písmeno po písmenu. Popřemýšlejte, zdali je opravdu nutné použít funkci `len`. Podmínkami zvyšujte hodnotu proměnných samohlásky a souhlásky o 1, když se daná hláska objeví ve slově. Na konci vypište, kolik je ve slově samohlásek a souhlásek. Program s komentáři je součástí přílohy č.1, soubor s názvem *priklad_9-cyklus_for.py*.

## 4.7 Seznamy

Seznam je typ objektů, pro který se používá v angličtině označení list. Práce s nimi je možná i v jazyce Scratch, pod stejným označením. V Pythonu je seznam dynamická datová struktura, z čehož vyplývá, že nemá statickou velikost. Slouží především k uchovávání většího množství dat. Oproti řetězcům nemají seznamy předem určený datový typ, takže v nich můžeme kombinovat čísla, číselné výrazy nebo písmena. K jednotlivým prvkům v seznamu se přistupuje pomocí indexů. Seznam je v podstatě něco jako třídní kniha, ve které má každý žák svoje číslo neboli index, pod tímto číslem se už ukrývá konkrétní osoba neboli prvek seznamu. Pokud potřebujeme do seznamu uložit sekvenci dat, lze použít funkci `range()`, o které již byla zmínka v kapitole o cyklu for [4, 14 ,18].

**Ukázka vytvoření seznamu:**

```
seznam = ["""první prvek seznamu""", """druhý prvek seznamu""", """ntý
prvek seznamu"""]
```

Po vytvoření seznamu s ním můžeme provádět operace, jako je změna pořadí prvků, nahrazování prvků, přidávání na konec seznamu, volání jednotlivých prvků seznamu atd. V tabulce 2 jsou popsány jednotlivé základní operace se seznamy, které můžete prakticky využít.

*Tabulka 2 Rozhraní seznamů a funkce pro práci s nimi*

| Operace | Význam operace |
|---|---|
| `seznam.append(prvek)` | do seznamu se přidá prvek na poslední místo |
| `seznam.insert(i, prvek)` | vloží prvek na danou pozici |
| `seznam.extend(sekvence)` | do seznamu se přidá sekvence prvků |
| `seznam.remove(hodnota)` | odstraní ze seznamu daný prvek |
| `seznam.pop()` | odstraní poslední prvek v seznamu |
| `seznam.clear()` | odstraní všechny položky seznamu |
| `seznam.reverse()` | vrací prvky seznamu v obráceném pořadí |
| `seznam.count()` | vrací počet výskytů daného prvku v seznamu |
| `seznam.sort()` | seřadí seznam podle abecedy, čísla podle velikosti |
| `len(seznam)` | vrací délku seznamu |
| `min(seznam)` | vrací nejmenší prvek v seznamu |
| `max(seznam)` | vrací největší prvek v seznamu |
| `sum(seznam)` | vrací součet všech hodnot v seznamu |

(Zdroj: vlastní zpracování na základě literatury [4, 14])

**Příklad číslo 10:**

Vytvořte program, který bude počítat průměr, počet jedniček a pětek v daném předmětu. Vstupy budou název předmětu a pomocí „nekonečného“ cyklu načtené známky, tyto známky budou uloženy do seznamu. Zadávání známek bude ukončeno mezerníkem, tj. načtením znaku mezery. Pomocí funkcí seznamu vypište počet výskytů jedniček, pětek a vypočítejte jeho aritmetický průměr. Zájemci mohou vložit podmínku, která v případě, že bude pětek více než jedniček, vypíše hlášku „Je potřeba zabrat!“. Program s komentáři je součástí přílohy č.1, soubor s názvem *priklad_10-seznamy.py*.

# 5 Úlohy inspirované učebnicemi

V této kapitole bude rozpracováno několik úloh, které mohou být inspirovány učebnicemi pro jazyk Scratch, nebo mohou být prakticky využívány ve skutečném světe. Tato kapitola je zde začleněna především proto, aby pomohla vytvořit propojení jednotlivých konstrukcí a principů, které jsou již žákům známy z jazyka Scratch do jazyka Python. Tím bude docíleno, že žáci nebudou vstřebávat novou sémantiku, ale pouze syntaxi, a tím si žáci vytvoří ucelenou představu, kdy každý grafický blok má za sebou blok příkazů v textovém rozhraní. Budu se snažit vybírat takové úlohy, které nebudou náročné z pohledu syntaxe a zároveň aby se co nejvíce podobaly základním konstruktům, které jsou rozepsané v podkapitolách kapitoly číslo čtyři. Každá úloha bude obsahovat název daného úkolu, pokyny pro vypracování, očekáváný výstup a napsaný kód v Pythonu. U cvičení, které jsou velkou měrou inspirována učebnicemi, bude i ukázka realizace v blokovém programovacím jazyce.

## 5.1 Program na procvičení malé násobilky

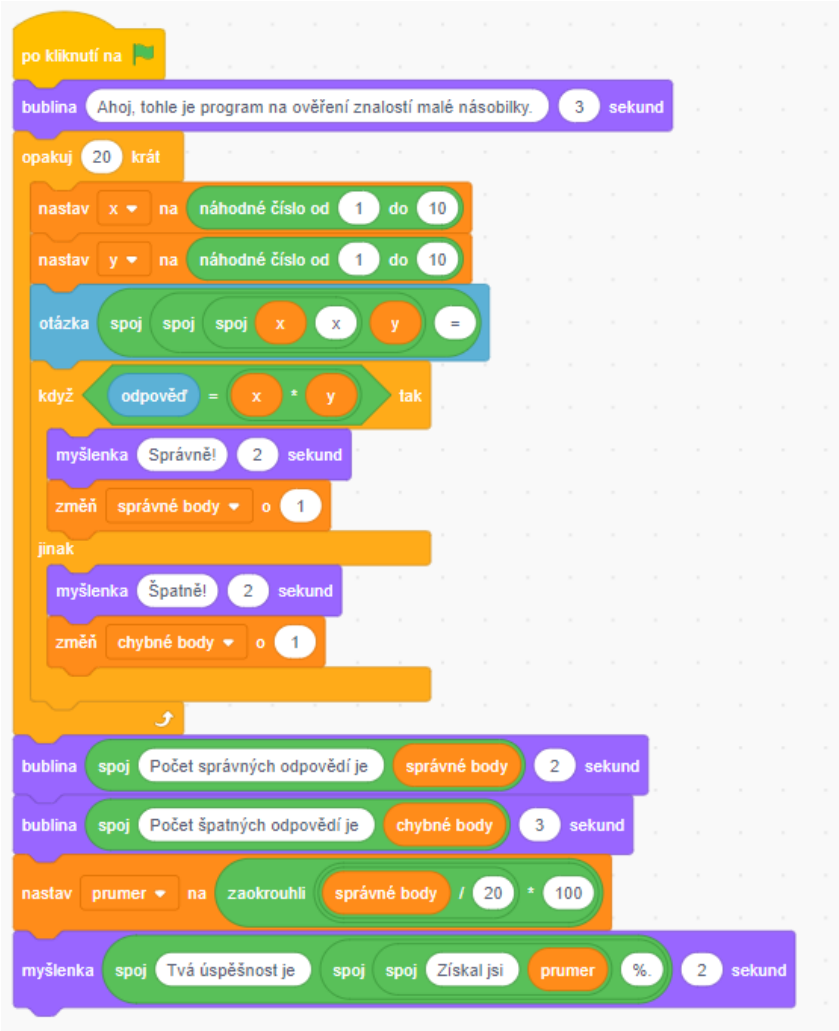
**Pokyny pro vypracování úlohy:**

V jazyce Python naprogramujte hru, která bude sloužit k procvičení malé násobilky. Na pozadí se budou generovat dvě náhodná čísla v rozsahu 1 až 10. Následně se zobrazí napsaný příklad násobení těchto dvou náhodných čísel a program bude chtít výsledek. V případě, že hráč zadá správný výsledek přičte se mu 1 bod, v případě, že jeho odpověď bude špatná, přičte se jeden bod chybový. Takto se bude opakovat 20 příkladů, kdy se s koncem hry vypíše počet správných a špatných odpovědí a procentuální úspěšnost. Využití této aplikace je pro všechny, kteří si chtějí ověřit znalosti malé násobilky. Prostou změnou parametrů se může cvičit i velká násobilka [20].

**Očekáváný výstup:**

V tomto příkladě žáci aplikují osvojené znalosti používání cyklů, definici proměnných, tvorbu podmínek a výpis na obrazovku. Žáci provedou analýzu problému, navrhnou řešení a následně ho naprogramují. V závěru prezentují své výsledky a poznatky.

**Možné řešení v blokovém jazyce:**




*Obrázek 7 Ukázka možného řešení v blokovém jazyce Scratch*

(Zdroj: vlastní zpracování)

Program v Pythonu s komentáři je součástí přílohy č.1, soubor s názvem *priklad_11-nasobilka.py*, program v jazyce Scratch je v souboru s názvem *priklad_11-nasobilka.sb3*.

## 5.2 Želví grafika kreslení tvarů

**Pokyny pro vypracování úlohy:**

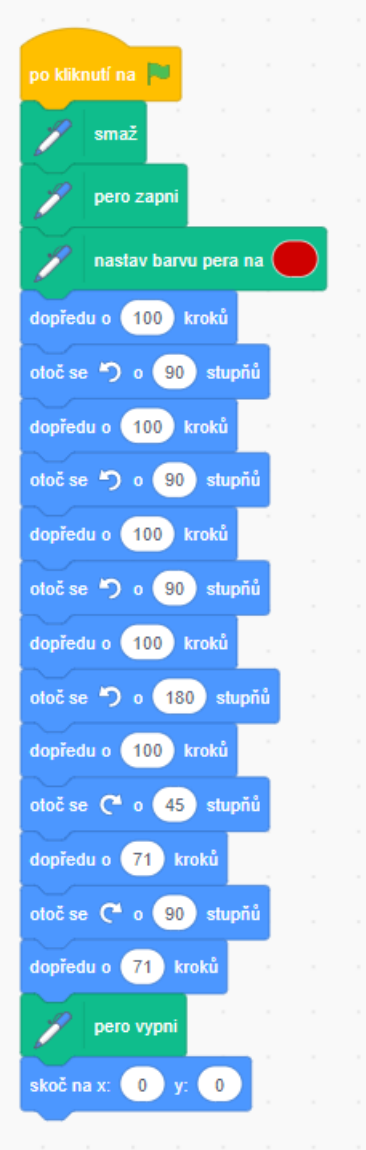
Podobně jako v jazyce Scratch, tak i v Pythonu existuje možnost kreslení. Tu obstarává tzv. želví grafika, kterou si naimportujeme jako knihovnu `from turtle import *`. Příkazem `forward` určujeme počet pixelů, o které se objekt posune dopředu. příkazy `left`, `right` a jejich hodnotou měníme úhel natočení. Při příkazu `left` je úhel 90° nahoru, 180° vlevo, 270°

dolů, 360° vpravo. Zadáním úkolu je, abyste vykreslili domeček pomocí želví grafiky. Barvu volte na základě osobních preferencí. Případní zájemci mohou nakreslit srdce [20].

**Očekáváný výstup:**

V tomto příkladě žáci aplikují osvojené znalosti používání želví grafiky ve Scratchi. Jako pomoc při přípravě a analýze problému si může žák narýsovat objekt na papír. Následně přenese své řešení do programovacího jazyka. Náplní tohoto cvičení není programování, ale spíše procvičení matematiky, představivosti a umění spojit geometrii s počítačem.

**Možné řešení v blokovém jazyce:**




*Obrázek 8 Kreslení domečku v blokovém programovacím jazyce*

(zdroj: vlastní zpracování)

Programy v Pythonu s komentáři jsou součástí přílohy č.1, soubory s názvem *priklad_12-kresleni_domu.py*, *priklad_12-kresleni_srdce.py*, program v jazyce Scratch je v souboru s názvem *priklad_12-kresleni_domu.sb3*.

## 5.3 Nákupní seznam

**Pokyny pro vypracování:**

Maminka Vás pošle na nákup. Jelikož téměř vždy na něco zapomenete, vytvořte si program, pomocí kterého se všechny položky nákupního seznamu zapíší do souboru. Pro zápis a čtení souborů jsou již v souboru předpřipravené funkce, které vám usnadní práci. Vašim úkolem bude doplnit zdrojový kód o nekonečný cyklus, pomocí kterého budete přidávat položky do seznamu. Jakmile budete mít všechno poznačené, pomocí mezerníku se provede zápis do souboru.

**Očekáváný výstup:**

V tomto příkladě žáci aplikují znalosti pro práci s cykly, podmínkami a práci se seznamem. Pomocným materiálem může být kapitola 4.7 a přiložený úkol. Nad rámec tohoto se žáci seznámí s prací se soubory, uvidí její syntaxi, čímž můžou rozvíjet svůj zájem v této oblasti, která je vstupní branou k datům ve formátu CSV.
Program s komentáři je součástí přílohy č.1, soubor s názvem *priklad_13-nakupni_seznam.py*.

## 5.4 Kdy budou prázdniny

**Pokyny pro vypracování:**

Každý z nás už se těší, až budou nějaké prázdniny. Proto, abychom to nemuseli složitě počítat, můžeme využít programování. Pro práci s časem slouží knihovny `time` a `datetime`. Do předpřipraveného programu deklarujte vstupy, které si vyžádají od uživatele číselné označení měsíce a dne prázdnin. V pomocné proměnné vypočítejte rozdíl dvou dat, konkrétní požadovaný datum a datum prázdnin, nezapomeňte dát rozdíl do závorek, za které umístíte tečku a připíšete klíčové slovo `days`. Spusťte program a ověřte jeho funkčnost.

**Očekáváný výstup:**

Žáci si vyzkoušejí práci s knihovnami obsluhující operace s časem.
Program s komentáři je součástí přílohy č.1, soubor s názvem *priklad_14-prace_s_casem.py*.

## 5.5 Převod stupňů Celsia na Kelviny a Fahrenheita

**Pokyny pro vypracování:**

Napište program, který pomocí vzorců bude přepočítávat hodnotu ze stupňů Celsia na Fahrenheita a Kelviny. Od uživatele si vyžádejte počáteční hodnotu, koncovou hodnotu a přírůstek, který určí po kolika stupních se bude počítat. Poté pomocí cyklu vypočítejte a vypište hodnoty pro daný rozsah. Pamatujte, že v Pythonu se ve vzorci používá desetinná tečka.

**Očekáváný výstup:**

Tento příklad je opakováním cyklu `for`, vyjádření matematických vzorů a ovládání parametrů funkce `range`. Může sloužit pro převod jednotek ve fyzice.

Program s komentáři je součástí přílohy č.1, soubor s názvem *priklad_15-prevod_stupnu_celsia_na_farhenheita_a_kelvina.py*.

## 5.6 Využití AI a již hotových programů ke tvorbě GUI

**Pokyny pro vypracování:**

Pomocí nástroje umělé inteligence například ChatGPT vytvořte k již existujícímu programu GUI neboli grafické uživatelské prostředí. Vygenerovaný kód zkuste spustit, odladit jeho chyby a popsat jednotlivé známé funkce v kódu. V závěru samotný kód otestujte a prodiskutujte se spolužáky problémy, se kterými jste se setkali.

**Očekáváný výstup:**

Tento příklad je zaměřený na využít umělé inteligence ve výuce programování. AI jak je zkráceně umělá inteligence označovaná, dokáže generovat rozumné a funkční programy. Žáci si tímto procvičí formulování dotazů a využití těchto nástrojů. Analýzou programu, který napsal někdo jiný si žáci vyzkouší práci hledání chyby, která je v programování velice důležitá. Procvičí si tvorbu popisků částí programů, pro případ, že by jejich kód dostal někdo jiný. Ve formě diskuse provedou obhájení svých závěrů, porovnají svá řešení a pokusí se najít podobnosti a odlišnosti v programech generovaných umělou inteligencí.

Program s komentáři je součástí přílohy č.1, soubor s názvem *priklad_16-grafika_s_vyuzitim_ai.py*.

## 5.7 Hrací kostka

**Pokyny pro vypracování:**

Do předpřipraveného souboru naimportujte knihovnu pro náhodné číslo `random`. Do předpřipravené funkce `hod()` si do pomocné proměnné, kterou pojmenujete podle svého uvážení vytvořte generování náhodného čísla v rozsahu hrací kostky. Na následující řádek, do závorek k argumentu `text` vložte formátovací řetězec obsahující text a pomocnou proměnnou s náhodným číslem. Poté aplikaci spusťte a otestujte.

**Očekávaný výstup:**

V rámci tohoto příkladu žáci aplikuji osvojené znalosti o formátovacím řetězci a náhodném čísle. Nad rámec toho metodou pozorování uvidí tvorbu grafického prostředí v knihovně `Tkinter`. Každé tlačítko, text nebo zadávání textu je odborně označováno jako widget. V příkladu je použita metoda `grid`, která zarovnává widgety do tabulky. Program s komentáři je součástí přílohy č.1, soubor s názvem *priklad_17-hraci_kostka.py*.

## 5.8 Kalkulačka

**Pokyny pro vypracování:**

V předpřipraveném programu se nachází program pro kalkulačku. Toto cvičení obsahuje práci s knihovnami `tkinter` a `math`. Analýzou zdrojového kódu žáci rozpoznají konstrukce, které jsou popsány v této bakalářské práci. K těmto konstrukcím vytvoří popisky. V případě, že se setkají s něčím, co není zahrnuto v této práci využijí prostředky internetu k vyhledání definic a tvorbě popisu.

**Očekáváný výstup:**

Žáci popíší jednotlivé části programu. Tyto závěry konzultují ve skupině. Pozorováním a aplikací probraného učiva žáci rozeznají definici funkce, globální a lokální proměnné, metodu formátování `grid` a tvorbu grafického rozhraní. Program je součástí přílohy č.1, soubor s názvem *priklad_18-kalkulator.py*. Toto cvičení nelze programovat ve Scratchi.

# 6 Závěr

Analýzou příslušných dokumentů [1, 2, 6] bylo zjištěno, že programování v jazyce Python je na základní škole přípustné pouze za předpokladu, že bude realizováno jako zájmové vzdělávání. Správnou právní formou tohoto vzdělávání je středisko volného času. Nabízí dostatek volného času a není vázáno na příslušný rámcový vzdělávací program. Jediným omezením může být v tomto případě ředitel, který může omezit dobu konání tohoto kroužku.

V teoretické části jsem se věnoval porovnání programovacích jazyků Python a Scratch, popsal vybrané konstrukce odpovídající zaměření této práce, poukázal na několik faktů, proč je Python vhodný jazyk pro výuku programování pro základní školy. Zabýval jsem se porovnáním jednotlivých vývojových prostředí, popsal podmínky za kterých lze jednotlivá použít a doporučil jeden komplexní uživatelsky přívětivý a pokročilý nástroj. Myslím, že právě tato část může pomoci pedagogům s výukou nové informatiky na SŠ a programování v rámci kroužku na ZŠ, protože volba vhodného prostředí nemusí být jednoduchá.

V praktické části jsou ukázky zápisu jednotlivých konstrukcí, která mají za úkol usnadnit jejich interpretaci pro žáky. V rámci procvičení těchto základních konstruktů je implementováno několik příkladů, které mohou být skvělou inspirací vyučování programování na základní škole. Tyto projekty zahrnují například výpočty lineárních rovnic, malou násobilku, určení počtu samohlásek v řetězci, výpočet průměru známek, nebo tvorbu nákupního seznamu. Okrajově zde zahrnuji práci s umělou inteligencí a jejím využitím při výuce programování, přesněji při hledání chyb v programu a diskuse nad vygenerovanými programy umělé inteligence.

Doufám, že tato bakalářská práce bude přínosná pro všechny, kteří budou chtít začít programovat v kroužcích na ZŠ. Může být vodítkem, jak udělat výuku zajímavou, interaktivní a využívat výpočetní prostředky aktivně ke zjednodušení denních rutin. Propojení s fyzikou, matematikou nebo anglickým jazykem se v rámci zájmového vzdělávání úplně nabízí. Jsem toho názoru, že právě tyto úkoly mohou v žácích vzbudit mnohem větší míru zájmu, chuti experimentovat a zároveň přidají elán do tvorby vlastních složitějších projektů tak, aby byly přínosné v praktické výuce a sloužily jako most k překlenutí se mezi jazykem Scratch a Python. Tato práce obsahuje sadu 18 příkladů, které jsou rozepsány na přibližně 330 řádcích kódu. Většinu těchto cvičení lze vytvořit v obou programovacích jazycích.

# Anotace

| | |
|---|---|
| **Jméno a příjmení:** | Miroslav Juriček, DiS |
| **Katedra:** | Katedra technické a informační výchovy |
| **Vedoucí práce:** | doc. RNDr. Petr Šaloun, Ph.D. |
| **Rok obhajoby:** | 2023 |

| | |
|---|---|
| **Název práce:** | Sada úloh pro programování v jazyce Python na ZŠ |
| **Název v angličtině:** | A set of tasks for programming in Python at primary school |
| **Anotace práce:** | Bakalářská práce pojednává o problematice výuky programování v Pythonu na základní škole. V teoretické části se zabývá možnou formou výuky, legislativními omezeními, výběrem vhodného vývojového prostředí a jejich porovnáním. V praktické části popisuje základní konstrukty daného jazyka, včetně zadání příkladů na procvičení. |
| **Klíčová slova:** | Programování, Python, Scratch, Rámcový vzdělávací program, zájmové vzdělávání. |
| **Anotace v angličtině:** | This bachelor thesis deals with the issue of teaching Python programming at elementary schools. The theoretical part focuses on possible forms of teaching, legislative limitations, selection of suitable development environments and their comparison. The practical part describes the basic constructs of the language, including exercises for practice. |
| **Klíčová slova v angličtině:** | Programming, Python, Scratch, Framework Education Programme, Interest Education |
| **Přílohy vázané v práci:** | |
| **Rozsah práce:** | stran |
| **Jazyk práce:** | čeština |

# Použité zdroje

[1] Nová informatika v RVP ZV, 2022 [Online]. Edu [cit 10.2.2023]. Dostupné z: https://revize.edu.cz/nova-informatika-v-rvp-zv

[2] 2005/74 Sb. Vyhláška o zájmovém vzdělávání, 2022 [Online]. Atre [cit. 13.2.2023]. Dostupné z: http://www.atre.cz/zakony/page0284.htm

[3] MAZÁČOVÁ Nataša (2014). *Vybrané problémy obecké didaktiky* [online]. Praha: Univerzita Karlova, Pedagogická fakulta [cit. 10.2.2023]. ISBN 978-80-7290-677-2. Dostupné z: http://www.vyzkum-mladez.cz/zprava/1434886741.pdf

[4] TRNEČKA Martin (2022). *Základy programování v python první část* [online]. Olomouc: Univerzita Palackého, Přírodovědecká fakulta [cit. 16.2.2023]. ISBN neuvedeno. Dostupné z: http://trnecka.inf.upol.cz/teaching/skripta-zpp1.pdf

[5] MARJI, Majed (2014). *Learn to Program with Scratch* [online]. San Francisco, California: No Starch Press [cit. 16.2.2023]. ISBN 978159327543. Dostupné z: https://www.pdfdrive.com/learn-to-program-with-scratch-e19585569.html

[6] *Rámcový vzdělávací program pro obor vzdělání 26-41-M/01 Elektrotechnika* [online]**.** Praha: MŠMT**,** [cit. 2023-02-18]. Dostupné z: https://www.edu.cz/rvp-ramcove-vzdelavaci-programy/ramcove-vzdelavaci-programy-stredniho-odborneho-vzdelavani-rvp-sov/obory-l-a-m/26-elektrotechnika-telekomunikacni-a-vypocetni-technika/

[7] MYŠKA, Karel a Michal MUNZAR (2014). *Základní hardware a software, operační systém* [online]. Hradec Králové: Gaudeamus[cit. 23.2.2023]. ISBN 978-80-7435-456-4. Dostupné z: https://www.uhk.cz/file/edee/filozoficka-fakulta/studium/myska_-_zakladni_hardware_a_software_operacni_system.pdf

[8] IDLE. Python Software Foundation [online]. 2019 [cit. 7.3.2023]. Dostupné z: https://docs.python.org/3/library/idle.html

[9] HOFMAN, Martin. *Tvorba elektronického výukového kurzu se zaměřením na vývoj Android aplikací v Python* [online]. Plzeň, 2019 [cit. 11.3.2023]. Dostupné z: https://dspace5.zcu.cz/bitstream/11025/37709/1/DP_Hofman.pdf. Diplomová práce. Západočeská Univerzita v Plzni, Pedagogická fakulta. PhDr. Denis Mainz, Ph.D

[10] Instalation Guide. *JetBrains* [online]. [cit. 12.3.2023]. Dostupné z: https://www.jetbrains.com/help/pycharm/installation-guide.html

[11] Google Colaboratory. *Colab.research.google.com* [online]. 2023 [cit. 15.3.2023]. Dostupné z: https://colab.research.google.com/notebooks/welcome.ipynb

[12] CARBONNELLE, Pierre. Top IDE index. *PYPL* [online]. 2023 [cit. 15.3.2023]. Dostupné z: https://pypl.github.io/IDE.htm

[14] PILGRIM, Mark. *Ponořme se do Pythonu 3: Dive into Python 3* [online]. Praha, 2010 [cit. 18.3.2023]. Dostupné z: https://knihy.nic.cz/files/nic/edice/mark_pilgrim_dip3_ver3.pdf

[15] STEPHENSON, Ben. *The Python Workbook: A Brief introduction with exercises and solutions* [online]. 2014 [cit. 22.3.2023]. Dostupné z: https://www.pdfdrive.com/the-python-workbook-a-brief-introduction-with-exercises-and-solutions-d157778622.html

[16] DOWNEY, Allen. *How to think like a computer scientist: learning with python* [online]. Wellesley, MA: Green Tea Press, 2002, [cit. 26.3.2023]. ISBN 0-9716775-0-6. Dostupné z: https://www.greenteapress.com/thinkpython/thinkCSpy.pdf

[17] HORNIK, Tomáš, Michal MUSÍLEK, Eva MILKOVÁ. *Didaktika programování* [online]. Skriptum UHK, 2019, [cit. 26.3.2023]. Dostupné z: https://imysleni.cz/images/vyukove_materialy/UHK_Didaktika_programovani.pdf

[18] LUTZ, Mark. *Learning Python 4th Edition* [online]. Beijing: O´Reilly 2009 [cit. 2.4.2023]. ISBN 978-0-596-15806-4. Dostupné z: https://cfm.ehu.es/ricardo/docs/python/Learning_Python.pdf

[19] TRNEČKA Martin (2022). *Základy programování v python druhá část* [online]. Olomouc: Univerzita Palackého, Přírodovědecká fakulta [cit. 2.4.2023]. ISBN neuvedeno. Dostupné z: http://trnecka.inf.upol.cz/teaching/skripta-zpp2.pdf

[20] DRÁBKOVÁ Jindra (2019). Didaktika programování [online]. Skriptum TUL, 2019, [cit. 8.4.2023]. ISBN neuvedeno. Dostupné z: https://imysleni.cz/images/vyukove_materialy/TUL_Didaktika_programovani.pdf

# Seznam obrázků a tabulek

# Seznam zkratek

RVP – rámcový vzdělávací program

IT – informační technologie

PC – personal computer (osobní počítač)

MŠMT – Ministerstvo školství, mládeže a tělovýchovy

ŠVP – školní vzdělávací program

BIOS – basic input-output system

IDE – Integrated development environment (vývojové prostředí)

True – výraz pro pravdu, pravdivý výrok

False – výraz pro nepravdu, nepravdivý výrok

GUI – Graphic user interface (grafické uživatelské prostředí)

AI – Artificial intelligence (umělá inteligence)

ZŠ – základní škola

SŠ – střední škola

# Seznam příloh

**Příloha č. 1** ZIP archiv, obsahuje:

- tuto bakalářskou práci ve formátu pdf,
- zdrojové kódy k jednotlivým cvičením včetně komentářů.