Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Jazyk Python jako skriptovací jazyk pro Autodesk Maya

Bakalářská práce

2023 Lucie Scholzová

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Lucie Scholzová**
Osobní číslo:	**I20154**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Jazyk Python jako skriptovací jazyk pro Autodesk Maya**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

V teoretické části bakalářské práce bude seznámení se skriptovacím jazykem Python. Úvodní část práce nás seznámí se základy tohoto jazyka (proměnné, datové typy, operátory, příkazy, řídící struktury, funkce, atd.). Dále budeme seznámeni se skriptovacím rozhraním v Mayi a s programováním grafického uživatelského rozhraní. Další kapitoly budou věnovány samotnému skriptování a psaní zásuvných modulů.

V implementační části bude vytvořena webová prezentace práce za pomocí jazyka HTML5 a CSS stylů, ve smyslu interaktivního kurzu výuky skriptování v programu Autodesk Maya. Práce nás seznámí se skriptovacím jazykem Python pro Autodesk Maya a bude obsahovat články z teoretické části bakalářské práce a praktické návody v rozsahu cca 10 lekcí. Návody budou názorně ukazovat jednotlivé postupy, tak aby byly přehledné i pro začínající uživatele a doplněné o multimediální ukázky.

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

GALANAKIS, Robert. Practical Maya Programming with Python. Packt Publishing Ltd, 2014. ISBN 978--1-84969-472-8.

MECHTLEY, Adam a Ryan TROWBRIDGE. Maya Python for Games and Film: A Complete Reference for the Maya Python and the Maya Python API. Waltham: Morgan Kaufmann, 2012. ISBN 978-0-12-378578-7.

Autodesk Maya 2022 Help: Scripting [online]. Autodesk Inc., 2022 [cit. 2022-10-20]. Dostupné z: https://help.autodesk.com/view/MAYAUL/2022/ENU/?guid=GUID-1C60DBC0-002C-4035-ADC7--97AD2F390190.

Vedoucí bakalářské práce: **Ing. Zbyněk Kopecký**
Katedra informačních technologií

Datum zadání bakalářské práce: **16. prosince 2022**
Termín odevzdání bakalářské práce: **12. května 2023**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2023

Prohlašuji:

Práci s názvem Jazyk Python jako skriptovací jazyk pro Autodesk Maya jsem vypracovala samostatně. Veškeré literární prameny a informace, které jsem v práci využila jsou uvedeny v seznamu použité literatury.

Byla jsem seznámena s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 09. 05. 2023 Scholzová Lucie

ANOTACE

Bakalářská práce "Python jako skriptovací jazyk pro Autodesk Maya" poskytuje přehled o jazyce Python a jeho použití v Autodesk Maya. Tato práce se zabývá základy Pythonu a jeho aplikací v Maye. Poskytuje také krátký úvod do API Autodesk Maya, než se ponoří do specifik používání Pythonu v Maya. Nakonec se práce zabývá využitím Pythonu jako skriptovacího jazyka v Autodesk Maya.

KLÍČOVÁ SLOVA

Python, Maya, skriptovací jazyk, Autodesk, API Maya

TITLE

Python as scripting language for Autodesk Maya

ANNOTATION

The bachelor’s thesis "Python as a scripting language for Autodesk Maya" provides an overview of the Python language and its use in Autodesk Maya. This thesis deals with the basics of Python and its applications in Maya. It also provides a brief introduction to the Autodesk Maya API before diving into the specifics of using Python in Maya. Finally, the thesis explores the use of Python as a scripting language in Autodesk Maya.

KEYWORDS

Python, Maya, scripting language, Autodesk, Maya API

Obsah

Seznam zdrojových kódů

Seznam obrázků

Terminologie

Devkit, zkratka pro "Developer's Kit", obvykle označuje sadu nástrojů, dokumentace a příkladů, které jsou poskytovány vývojářům pro podporu vývoje aplikací a pluginů pro daný software nebo platformu.

Příznak se v Maya dokumentaci používá k označení jednoho z argumentů funkce nebo metody, který určuje, jak se má daná funkce nebo metoda chovat. Tyto příznaky jsou obvykle předávány ve formě textových řetězců. Často se zde používají příznaky pro nastavení různých vlastností objektů, výstupů a funkcí.

Syntaxe označuje pravidla, které určují strukturu a způsob zápisu programovacího jazyka nebo jiného formálního jazyka. Správná syntaxe je důležitá pro to, aby mohl být kód rozpoznán a interpretován strojem nebo jiným programem.

Proměnná je označení pro identifikátor, který slouží k ukládání a manipulaci s hodnotami v počítačovém programu. Proměnná může být pojmenována libovolným názvem a může obsahovat hodnoty různých datových typů, jako jsou čísla, řetězce, seznamy a další.

Metoda je funkce nebo procedura, která je součástí třídy v objektově orientovaném programování a slouží k manipulaci s vlastnostmi dané třídy. Metody umožňují vytvářet objekty s určitými funkcemi a chováním a jsou klíčové pro objektově orientované programování.

Responzibilní design zahrnuje pravidla a techniky pro vytváření přizpůsobivého uživatelského rozhraní, které dokáže správně zobrazovat obsah na různých zařízeních. Tím zajišťuje, že aplikace bude použitelná a přístupná pro co nejširší spektrum uživatelů.

Seznam zkratek

API	Aplikační programovací rozhraní
CSS	Kaskádové styly
GUI	Grafické uživatelské rozhraní
HTML	Hypertextový značkovací jazyk
IDE	Integrované vývojové prostředí
OOP	Objektově orientované programování

Úvod

Tématem této práce je využití jazyka Python jako skriptovacího jazyka pro Autodesk Maya. Práce je dělena do dvou částí. V teoretické části budeme seznámeni se skriptovacím jazykem Python. Úvodní část obsahuje základy tohoto jazyka, jako jsou proměnné, datové typy, operátory, příkazy, řídící struktury, funkce a podobně. Následovat bude skriptovací rozhraní v Maya a programování grafického uživatelského rozhraní. Poslední kapitola teoretické části bude věnována samotnému skriptování a psaní zásuvných modulů.

V rámci praktické části bude vytvořena webová prezentace. Prezentace bude navržena ve smyslu interaktivního kurzu výuky skriptování v programu Autodesk Maya. Práce seznámí účastníka kurzu se skriptovacím jazykem Python pro Autodesk Maya a bude obsahovat články z teoretické části této práce a praktické návod. Kurz bude obsahovat 10 lekcí.

1. Python

Python patří mezi univerzální moderní programovací jazyky. Umožňuje navrhovat jednoduché programy, ale zároveň poskytuje dostatečné prostředky pro navržení poměrně náročných programů. Řadí se mezi skriptovací jazyky, ve kterých je možné zapisovat skripty. Je vyvíjen jako open source projekt a umožňuje instalaci balíčků pro většinu běžných platforem, jako jsou Unix, Microsoft Windows, macOS a Android. Jednou z předností Pythonu je možnost využití relativně každého vývojového prostředí pro psaní skriptů. Nejvyužívanějším prostředím je prostředí IDE, Integrated Development Environment neboli integrované vývojové prostředí.

Nejnovější verzí jazyka Python, v době psaní této práce, je verze 3. Nová řada není kompatibilní s řadou 2.x. Liší se především funkčností vestavěných objektů, jako jsou slovníky a řetězce.

1.1. Syntaxe

Syntaxe Pythonu je velmi jednoduchá. Konce řádků nejsou ukončovány středníkem, jak to u většiny jazyků bývá, ale pouze odřádkováním. Bloky jsou určovány odsazením, standardně čtyřmi mezerami. Potřebná je alespoň jedna.

1.2. Proměnné

Python nemá pro deklaraci proměnné žádný přiřazený příkaz. Každá proměnná má svůj identifikátor, který slouží k její identifikaci. V Pythonu jsou pro identifikátory následující pravidla:

- Smějí obsahovat písmena, číslice a podtržítka.
- Nesmějí začínat číslicí.
- Rozlišují se malá a velká písmena.
- Nesmějí být shodné s klíčovými slovy jako jsou False, None, True, and a podobně.

Proměnné se vytvářejí přiřazením hodnoty, nemusí mít deklarovaný žádný typ a mohou svůj typ v průběhu měnit.

1.3. Datové typy

V jazyce Python existuje několik základních datových typů. Mezi základní patří celá čísla, desetinná čísla, řetězce, booleovské hodnoty a seznamy. Python také umožňuje definovat vlastní datové typy pomocí tříd. Datové typy v Pythonu jsou dynamické, to znamená, že typ proměnné může být automaticky určen.

1.3.1. Celá čísla

V Pythonu existují dva základní typy, int a long. Typ int může uchovávat celá čísla, jejichž velikost je omezena pouze pamětí počítače. Typ long umožňuje uchovávat libovolně velká celá čísla a umožňuje rozšíření předponou 0x pro hexadecimální zápis a 0o pro oktální zápis. S celými čísly můžeme provádět operace, jako je sčítání, odčítání, násobení, dělení, umocnění a modulo. Python obsahuje vestavěné funkce pro práci s celými čísly. Mezi nejpoužívanější patří abs(x), který vrátí absolutní hodnotu čísla x, a round(x, n) vracející číslo x zaokrouhlený na n celočíselných cifer. Dále funkce bin(i) vrátí binární reprezentaci čísla i ve formě řetězce a hex(i) hexadecimální reprezentaci čísla.

1.3.2. Logické hodnoty

Existují dvě Booleovské hodnoty True a False. Tyto hodnoty jsou výsledkem porovnávacích operací, kterými jsou rovnost, nerovnost, větší než, menší než, větší nebo rovno a menší nebo rovno. Logické hodnoty jsou využívány především v podmínkových výrazech, kde určují, zda se má určitá akce provést.

Dále se v Pythonu vyskytují logické operátory and, or a not, díky kterým je umožněna kombinace logických hodnot. And vrací True, pokud jsou oba výrazy True a not, vrací opačnou hodnotu daného výrazu.

Namísto True a False se v dřívějších letech pro porovnávání užívaly hodnoty 1 a 0.

1.3.3. Typy s pohyblivou desetinnou čárkou

V Pythonu existují tři typy hodnot s pohyblivou desetinnou čárkou. Prvním typem je typ float. Tento typ má omezenou přesnost a nelze ho spolehlivě porovnávat. Jsou zapisovány s desetinnou tečkou (např. 2.5, -1.2) nebo pomocí exponenciální notace (např. -2e9, 9.8e-4).

Dalším typem je decimal, jinak zvaný desetinné číslo, který umožňuje přesnou aritmetiku s desetinnými čísly. Tento typ je přesnější než float. Typ decimal je využíván v oblastech, kde je vyžadována větší přesnost ve výpočtech.

Posledním typem je complex, také nazývaný jako komplexní číslo. Tento typ uchovává dvojici čísel typu float, kdy jedno reprezentuje reálnou a druhé imaginární složku komplexního čísla. Zapisují se ve tvaru a + bj, kde j reprezentuje imaginární jednotku.

Typ float a complex můžeme mezi sebou kombinovat. Výsledkem této kombinace bude hodnota typu complex. Čísla typu decimal mají fixní přesnost, a proto je nelze zkombinovat s jiným datovým typem.

1.3.4. Řetězce

Řetězce se v Pythonu zapisují uvozovkami jednoduchými ‚ nebo dvojitými „“. Uchovávají posloupnost znakové sady Unicode. Tyto uvozovky jsou ekvivalentní, takže nezáleží na tom, jakou formu zvolíme. Chceme-li ale použít uvozovky v řetězci, je potřeba zvolit opačný typ, jako například v případě: "Toto je text ve dvojitých uvozovkách a řetězec 'uvnitř' něho.". Řetězce se v Pythonu po vytvoření nemohou modifikovat. Můžeme ale vytvořit nový řetězec na základě předchozího řetězce. Řetězce spojujeme operátorem +, operátorem * je opakujeme a můžeme je také porovnávat a řadit.

V Pythonu existuje mnoho vestavěných funkcí pro práci s řetězci. Mezi základní patří funkce len(), která vrací délku řetězce, metoda split(), která rozdělí řetězec na podřetězce podle zadaného oddělovače, a metoda join(), která spojí seznam řetězců.

Pomocí speciálních řetězcových sekvencí, které začínají zpětným lomítkem, můžeme například v případě \n oddělit nový řádek a v případě \t vložit tabulátor.

Pro řetězce můžeme používat také některé z předdefinovaných metod. Metoda S.capitalize() vrátí kopii řetězce, kde první písmeno bude převedeno na velké, S.count(t, začátek, konec) vrátí počet výskytů řetězce t v řetězci S, nebo v rozmezí začátek až konec. Metoda S.encode() vrací objekt typu bytes, S.find(t, začátek, konec) nachází nejlevější pozici t v řetězci S a S.format(…) generuje kopii řetězce S naformátovanou podle zadaných argumentů. [2] V Pythonu se vyskytuje na desítky dalších podobných funkcí.

1.4. Operátory

Operátor v Pythonu je znak, skupina znaků nebo skupina slov, které označují typ operace, která se má provést. Operátory dělíme na aritmetické, logické, srovnávací, bitové, operátory přiřazení, operátory příslušnosti a operátor identita. V následující kapitole si ale přiblížíme pouze operátory logické, aritmetické, bitové a operátory porovnávání.

1.4.1. Logické operátory

V Pythonu existují tři logické operátory, a to and, or a not. Operátory and a or používají logiku zkráceného vyhodnocení. Jejich výsledkem je operand, který rozhodl o výsledku. Pro představu si uvedeme příklad. Máme hodnotu five = 5 a two = 2. V případě použití and „five and two“ bude výsledkem operace 2. Kdybychom operandy otočili „two and five“ bude výsledkem 5. V případě použití or bude při stejných proměnných „five or two“ výsledek 5 a při „two or five“ výsledek 2. Pokud by se operátor použil v příkazu if, bude operace vyhodnocena jako logická hodnota s výsledkem True nebo False. Při použití and je výsledek kladný (True) v případě, že jsou oba operandy vyhodnoceny jako True. U operátoru or je výsledek kladný, pokud je alespoň jeden z použitých operandů vyhodnocen jako pravdivý.

Unární operátor not vrací vždy logickou hodnotu. V případě „not (five and two)“ by byl výsledek False.

1.4.2. Aritmetické operátory

Python nabízí čtyři základní operátory pro matematické operace: sčítání +, odčítání -, násobení * a dělení /. Python také nabízí operátory rozšířeného přiřazení, jako je += a *=. Operátor += může nahradit standardní + a tím zkrátit operaci. V případě existence proměnné a = 5 při operaci sčítání a = a + 2 můžeme příklad zkrátit pomocí operátoru na a += 2. Výsledek těchto operací bude totožný. U operátoru *= je použití stejné.

1.4.3. Bitové operátory

Bitové operátory se používají k porovnání binárních čísel. Existuje šest bitových operátorů. Základními operátory jsou & (and) pro bitovou konjunkci a | (or) pro bitovou disjunkci celých čísel. Dále existují operátory ^ (xor) pro bitovou nonekvivalenci, << operátor v případě i << j posune číslo i doleva o j bitů, >> operátor v případě i >> j posune číslo i o j bitů doprava a nakonec ~i obrátí bity čísla i. [2]

1.4.4. Operátory porovnávání

Tento typ operátoru porovnává dvě hodnoty. V Pythonu existují následující operátory porovnávání: == (rovná se), != (nerovná se), > (větší než), >= (větší nebo rovno), < (menší), <= (menší nebo rovno) a === (identita objektů, zda proměnné ukazují na stejné místo v paměti).

1.5. Datové struktury

Datové struktury jsou základním stavebním kamenem každého programovacího jazyka. Umožňují ukládat, organizovat a manipulovat s daty.

V této kapitole se zaměříme na vybrané datové struktury, jimiž jsou seznamy, n-tice, slovníky a množiny.

1.5.1. Seznamy

Seznam je jednou z nejpoužívanějších datových struktur v Pythonu. Je to dynamická struktura, což znamená, že může měnit svou velikost při běhu programu. Je to uspořádaná posloupnost hodnot libovolného datového typu. Seznamy jsou indexované a je možné k jednotlivým prvkům přistupovat pomocí hranatých závorek. Index značí pozici prvku a je číslován od nuly.

Prvky v seznamu je možné přidávat a odebírat pomocí metody S.append(x) a S.remove(x), kde S je seznam a x je vybraný prvek. Pro vložení prvku je také možné použít metodu S.inser(i, x), která vloží prvek x na indexovou pozici i v seznam S. Dalšími používanými metodami jsou S.count(x), který vrátí počet výskytů prvku x v seznamu S, S.sort(..), který seřadí seznam S, S.index(x) vracející indexovou pozici prvního výskytu prvku x zleva, S.pop() vracející a odstraňující nejpravější prvek a S.reverse(), jež se používá pro obrácení seznamu. Základní práce se seznamem je znázorněna v příkladu (Zdrojový kód 1).

```
>>> jmena = ["Jan", "Petr", "Marie", "Anna", "Lukáš"] # Vytvoření seznamu jmen
["Jan", "Petr", "Marie", "Anna", "Lukáš"]
>>> print(jmena[0]) # Vypíše: Jan # Výpis prvního jména v seznamu
Jan
>>> print(len(jmena)) # Vypíše: 5 # Výpis délky seznamu
5
>>> jmena.append("Kateřina") # Přidání jména na konec seznamu
["Jan", "Petr", "Marie", "Anna", "Lukáš", " Kateřina"]
>>> jmena.sort() # Seřazení seznamu podle abecedy
["Anna", "Jan", "Kateřina", "Lukáš", "Marie", "Petr"]
>>> print(jmena) # Výpis seřazeného seznamu
Anna Jan Kateřina Lukáš Marie Petr
```

Zdrojový kód 1 Seznam

1.5.2. N-tice

N-tice, jinak zvané tuple, jsou datové struktury sloužící k ukládání kolekcí hodnot podobně jako seznamy. Jsou to uspořádané posloupnosti nula nebo více odkazů

na objekty. Je to neměnná datová struktura, tudíž jejich stav není možné měnit po jejich vytvoření. V případě, že chceme n-tici změnit po vytvoření, je potřeba ji převést pomocí metody list() na seznam. N-tice je možné rozšiřovat pomocí operátorů přiřazení += a *=.

Jsou definované pomocí kulatých závorek. Nabízejí metody N.count(x), které vrátí počet výskytů x v n-tici N, a metodu N.index(x), která vrací pozici prvního výskytu x v n-tici. Je na ně možné aplikovat operátory + na spojení, * na replikaci a [] na řezání. Taktéž můžeme aplikovat operátory in a not pro testování příslušnosti.

```
>>> tuple = ("apple", "banana", "cherry") # vytvoření n-tice
("apple", "banana", "cherry")
>>> tuple.count("apple") # zjištění počtu výskytů „apple“
1
>>> tuple.index("banana") # zjištění prvního výskytu „banana“
1
>>> tuple1[1:3] # oříznutí n-tice
("banana", "cherry")
```

Zdrojový kód 2 N-tice

1.5.3. Slovníky

Slovníky slouží k ukládání párových hodnot nazývaných klíče a hodnoty. Každý klíč ve slovníku musí být unikátní. Jsou definovány pomocí složených závorek a jednotlivé klíče jsou odděleny dvojtečkou. Klíč může být řetězec, číslo nebo jiný datový typ, který je hashovatelný (nezměnitelný). Hodnoty mohou být libovolného datového typu, včetně slovníků.

Slovník je upravovatelná datová struktura a je možné položky přidávat pomocí hranatých závorek, do kterých se vloží klíč položky a následně je pomocí operátoru = přidána hodnota ke klíči. Je možné hodnoty mazat pomocí příkazu del. Také můžeme zjistit velikost slovníku pomocí metody len(). Tyto zmíněné operace jsou znázorněny v následující tabulce (Zdrojový kód 3).

```
>>> slovnik = {"Jablko": "Apple", "Knoflík": "Button", "Myš": "Mouse"} #
vytvoření slovníku
```

```
{"Jablko": "Apple", "Knoflík": "Button", "Myš": "Mouse"}
>>> slovnik["Jablko"] # zjištění hodnoty na indexu "Jablko"
"Apple"
>>> slovnik["Pes"] = "Dog" # Přidání položky do slovníku
{"Jablko": "Apple", "Knoflík": "Button", "Myš": "Mouse", "Pes": "Dog"}
>>> del slovnik['Pes'] # vymazání hodnoty ze slovníku
{"Jablko": "Apple", "Knoflík": "Button", "Myš": "Mouse"}
>>> len(slovnik) # Zjištění velikosti slovníku
3
```

Zdrojový kód 3 Slovníky

1.5.4. Množiny

Množiny neboli set jsou datové struktury, které ukládají jedinečné prvky, které se v množině neopakují. Množiny je možné vytvářet pomocí globální funkce set(). Druhým způsobem vytvoření je použití složených závorek. Jsou podobné seznamům ale na rozdíl od nich nemají pořadí.

Na množiny je možné aplikovat metodu len() pro zjištění velikosti množiny, add() pro přidání prvku, remove() pro odebrání a clear() pro odstranění všech položek v množině. Také je možné použít operátory in a not in k ověření, zda se prvek nachází v množině.

```
>>> planety = set(["Země", "Mars", "Jupiter", "Saturn", "Uran", "Neptun"])
{" Země ", " Mars ", " Jupiter ", " Saturn ", " Uran ", " Neptun "}
>>> staty = {"Česká republika", "Polsko", "Slovensko"}
{" Česká republika ", " Polsko ", " Slovensko "}
>>> planety.add("Pluto")
{" Země ", " Mars ", " Jupiter ", " Saturn ", " Uran ", " Neptun ", "Pluto"}
>>> panety.remove("Země")
{" Mars ", " Jupiter ", " Saturn ", " Uran ", " Neptun ", " Pluto "}
>>> planety.clear()
set()
```

Zdrojový kód 4 Množiny

1.6.Klíčové slovo pass

Klíčové slovo pass se používá v případech, kde je vyžadována přítomnost sady, ale z nějakého důvodu ji nechceme použít. Pass slouží jako nic nedělající operace. V případě jeho použití je sada přeskočena a pokračuje se dále k vyhodnocování mimo daný blok.

1.7. Řídící struktury

Řídící struktury jsou základní stavební kameny každého programu. Umožňují řídit chování našeho kódu. Jde o posloupnost příkazů, větvení a cyklení kódu.

Řídící struktury se dělí do dvou kategorií, a to na podmínky a cykly. Pro jejich vyhodnocení využíváme logické výrazy, které lze vyhodnotit na logickou hodnotu True nebo False. Pokud se jedná o předdefinovanou konstantu False, speciální objekt None, prázdnou posloupnost, kolekci nebo číselný datový prvek s hodnotou 0, pak takový výraz bude vyhodnocen jako False [2] V opačném případě je vyhodnocen jako True. V případě vytvoření vlastních datových struktur se můžeme rozhodnout, jaký má být výsledek.

Bloky jazyka Python vyžadují přítomnost sady neboli posloupnost jednoho či více příkazů, a z tohoto důvodu zde existuje klíčové slovo pass (1.6.).

1.7.1. Podmínky

Do kategorie podmínek se řadí pouze if. Je zapisována klíčovými slovy if, elif a else. Podmíněný výraz if umožňuje větvení. V případě, že je logický výraz v podmínce vyhodnocen jako True, je následující sada provedena. V opačném případě je segment přeskočen. V případě, že je podmínka následována příkazem else, je v případě předešlého vyhodnocení jako False vyhodnocena tato sada. Větev else je nepovinná a může být použita pouze jednou. V případě, že je ale if následován elif, je else povinný. Větev elif slouží pro vyhodnocení dalšího logického bloku a může jich být v řadě několik. Pokud je vyhodnocen jako True, je vykonán. V opačném případě se pokračuje do větve else.

Příkazy if a elif následuje logický výraz. Ten ale nemusí být, jak to u jiných jazyků bývá, oddělen závorkami. Příkaz else je následován pouze dvojtečkou.

```
>>> x = 6
>>> if x>5:
True
>>>>>> print("číslo je větší než 5")
číslo je větší než 5
>>> elif x==0:
False
>>>>>> prit("číslo je 0")
>>> else:
>>>>>> print("číslo je menší než 5 a není 0")
```

Zdrojový kód 5 Podmínky

1.7.2. Cykly

V Pythonu existují dva typy cyklů, while a for. Cyklus typu for se používá pro iteraci přes prvky v nějaké sekvenci nebo kolekci dat a cyklus while se používá v případech, kdy je potřeba opakovat určitý blok do splnění dané podmínky.

1.7.2.1. Cyklus while

```
>>> while logicky_vyraz:
>>>>>> sada_while
>>> else:
>>>>>> sada_else
```

Zdrojový kód 6 Definice While

Cyklus while se využívá v případech, kdy neznáme přesný počet opakování. z uvedeného zdrojového kódu (Zdrojový kód 6) vyplývá, že syntaxe cyklu while se skládá z logického výrazu (logicky_vyraz), který určuje, zda má cyklus probíhat, a z větve else, která je nepovinná a může obsahovat blok, který se má vykonat v případě, že logický výraz není True. V případě, že je v bloku while uveden příkaz continue, je ukončeno vyhodnocování bloku a řízení se vrátí na začátek cyklu a znovu

se vyhodnotí logický výraz. Pokud je cyklus ukončen příkazem break nebo return, anebo když je vyvolaná výjimka, blok else se neprovede.

Cyklus můžeme předčasně ukončit pomocí příkazů break a return. V případě použití return, musí být cyklus ve funkci či metodě.

Cyklus while se může jednoduše zacyklit v případě, že podmínka nikdy nebude splněna. K tomu dojde, když cyklus napíšeme například následujícím způsobem (Zdrojový kód 7).

```
>>> while True:
>>>>>> print("číslo")
```

Zdrojový kód 7 Nakonečný while

Z takového bloku se dostane pouze zrušením procesu anebo doplněním příkazu break po splnění vnitřní podmínky.

```
>>> cislo = 1
>>> while True:
>>>>>> print("číslo")
>>>>>> cislo += 1
>>>>>> if cislo == 5:
>>>>>>>>> break
```

Zdrojový kód 8 Použití break ve while

1.7.2.2. Cyklus for

Cyklus for využijeme v případě, že chceme provést předem daný počet opakování. Používá se především v procházení sekvencí. Sekvence jsou kontejnerové datové struktury. Jsou to proměnné, které mohou obsahovat více položek, jako jsou řetězce, seznamy a všechny objekty, které mají metodu __iter__. Cyklus projde všechny prvky v sekvenci a v každé iteraci se aktuální prvek uloží do definované proměnné.

```
>>> for prvek in sekvence:
>>>>>> sada_for
>>> else:
>>>>>> sada_else
```

Zdrojový kód 9 Definice for

V případě, že je uvnitř bloku sada_for proveden příkaz continue, vrátí se řízení, podobně jako u cyklu while, na začátek cyklu a spustí se nová iterace. Po dokončení všech iterací se provede případný blok else. V případě přerušení cyklu pomocí break, return nebo v případě vyvolání výjimky se blok else neprovede. V případě použití return, musí být cyklus ve funkci či metodě.

```
>>> sekvence = ["jedna","dva","tri"]
>>> for cislo in sekvence:
>>>>>> print(cislo)
```

Zdrojový kód 10 Cyklus for

Pro cyklus for můžeme využít funkce range() (Funkce range()).

```
>>> for i in range(1, 11):
>>>>>> print(i) # vypíše čísla od 1 do 10
```

Zdrojový kód 11 Použití funkce range() v cyklu for

Funkce range()

Funkce range() vrací vygenerovaná čísla v zadaném rozmezí. Funkce může mít tři parametry range(začátek, konec, krok). Začátek určuje, od kterého čísla generování začíná. Jedná se o nepovinný parametr. V případě, že není udán, automaticky je nastaven na 0. Konec udává koncovou číslovku rozmezí generovaných čísel, tento parametr je povinný. Poslední volitelný parametr je krok, který specifikuje způsob inkrementace a výchozí hodnota je 1.

```
>>> range(10) # vrátí čísla od 0 do 9
>>> range(5,10) # vrátí čísla od 5 do 9
>>> range (1,10,2) # vrátí čísla od 1 každé 2 číslo do 9
```

Zdrojový kód 12 Funkce range()

1.8. Výjimky

Výjimky využíváme v případech, kdy potřebujeme zachytit nějaký nežádoucí průběh kódu, špatný formát parametru či jiného vstupu nebo nežádoucí výsledek. V případě, že chceme na vyvolanou výjimku reagovat, je potřeba ji nejprve zachytit.

Zachytávání výjimek probíhá pomocí příkazu try/except, který je znázorněn v příkladu (Zdrojový kód 13).

```
>>> try:
>>>>>> blok_try
>>> except skupina_vyjimek as výjimka:
>>>>>> blok_except
>>> else:
>>>>>> blok_else
>>> finally:
>>>>>> blok_finaly
```

Zdrojový kód 13 Syntaxe zachycení výjimek

Alespoň jeden blok except je povinný. Může ho následovat libovolný počet dalších bloků except. Dalšími volitelnými bloky jsou else a finally. Blok else se provede při normálním dokončení bloku try, ale v případě vzniku výjimky se neprovede. Blok finally se provede vždy po dokončení.

Každý příkaz except může obsahovat jednu výjimku nebo skupinu výjimek uzavřených do závorek.

```
>>> try:
>>>>>>int("číslo")
```

```
>>> except ValueError:
>>>>>> print("chyba zachycena")
>>> except (TypeError, AttributeError):
>>>>>> print("vyskytla se jedna z uvedených chyb")
```

Zdrojový kód 14 Zachycení výjimky

Provede-li se blok try bez chyby, je blok except ignorován. V případě vyskytnutí očekávané výjimky se provede odpovídající blok except. V případě vyvolání jiné výjimky, než která je zadána v příkazu except, ukončí Python program a vypíše chybovou hlášku do konzole.

Výjimky také můžeme uměle vyvolat. Toho využijeme v případě, kdy potřebujeme změnit tok programu, například nechceme-li, aby vstup funkce byl nějaká konkrétní hodnota. Výjimky se vyvolávají pomocí příkazu raise. Příkaz následuje typ výjimky a do závorek se udává popis toho, co je třeba opravit. Příkaz raise ukončí funkci nebo program.

```
>>> if paramer > 0:
>>>>>>> print("správný formát parametru")
>>> else:
>>>>>>> raise ValueError("parametr musí být větší než 5")
```

Zdrojový kód 15 Vyvolání výjimky

1.9. Funkce

V Pythonu existuje několik druhů funkcí. Jsou to například lokální, globální, lambda funkce a metody.

Globální funkce jsou veškeré funkce, které jsou přístupné z jakékoliv části kódu ve stejném modulu. Je možné k nim přistupovat i z jiných modulů, a to pomocí importování daného modulu (1.11.).

Lokální funkce, někdy označované jako vnořené funkce, jsou definovány uvnitř jiných funkcí. Tyto funkce jsou viditelné pouze ve funkci, ve které jsou definované. [2]

Lambda funkce můžeme vytvářet pouze v okamžiku jejich použití.

Metody jsou funkce, které jsou spojené s určitým datovým typem. Budou více probrány v kapitole třídy (1.10.).

1.9.1. Lokální a globální funkce

Funkce v Pythonu definujeme pomocí příkazu def, za které je vloženo jméno funkce. Do závorek vložíme případné parametry a nakonec dvojtečku. V případě, že chceme učinit parametr nepovinným, udáme mu výchozí hodnotu. Výchozí hodnota se zadává pomocí rovná se a za ním vložené hodnoty (Zdrojový kód 17).

Na začátku těla funkce můžeme vložit popis, co funkce dělá, kterému se říká dokumentační řetězec, anglicky docstring, který se standardně označuje třemi uvozovkami.

```
>>> def prvniFunkce(parametry):
>>>>>> """ dokumentační řetězec """
>>>>>> pass
```

Zdrojový kód 16 Syntaxe lokální a globální funkce

V ukázce je obecná syntaxe pro vytvoření lokální či globální funkce. Je zde také ukázán dokumentační řetězec a použito klíčové slovo pass (1.6.).

Funkce mohou obsahovat příkaz return, pomocí kterého mohou vrátit libovolnou hodnotu. Funkce v Pythonu vždy vrací hodnotu. V případě, že není funkce ukončena příkazem return, automaticky vrací None.

V některých případech může být příkaz return nahrazen příkazem yield. Yield se používá v takzvaných generátorových funkcích, které slouží k vytváření sekvenčních hodnot, které jsou generovány postupně a "líně" (lazy), tedy až v okamžiku, kdy jsou požadovány.

```
>>> def vypocet(a, b , c = 1):
>>>>>> s = (a + b + c) / 2
>>>>>> return s
```

Zdrojový kód 17 Ukázka lokální či globální funkce

Volání funkce je uskutečněno pomocí zapsání jejího jména a umístěním parametrů do kulatých závorek. Pokud funkce parametry nemá, budou kulaté závorky prázdné.

```
>>> vysledek1 = vypocet(1, 2, 3) # a = 1, b = 2, c = 3
vysledek1 = 3
>>> vysledek2 = vypcoet(1, 2) # volání s využitím výchozí hodnoty a = 1, b = 2, c = 1
vysledek2 = 2
```

Zdrojový kód 18 Volání funkce

1.9.2. Lambda funkce

Funkce lambda jsou běžně označovány jako anonymní funkce. Mohou mít libovolný počet argumentů, ale pouze jeden výraz. Využívají se především tam, kde chceme vytvořit funkci, která bude obsahovat pouze jednoduché výrazy.

```
>>> lambda parametry: výraz
```

Zdrojový kód 19 Syntaxe lambda funkce

Lambda se vytváří podle syntaxe uvedené v ukázce (Zdrojový kód 20). Parametry jsou volitelné. Na rozdíl od funkcí, lambda nemá žádný název. Výraz nesmí obsahovat větvení ani cykly a nesmí mít příkaz return, nebo yield.

```
>>> vysledek = lambda x : "" if x == 1 else "s"
>>> print(vysledek(0))
```

Zdrojový kód 20 Příklad lambda funkce

V ukázce (Zdrojový kód 20) je uvedeno použití lambda funkce. Anonymní funkce má parametr x, na jehož základě je rozhodnuto o výsledku. Pokud je x roven 1, je výsledkem prázdný string, pokud je x cokoliv jiného, je výsledkem string s hodnotou s.

1.10. Třídy

Třídy jsou základním stavebním kamenem OOP a v Pythonu se používají k definici objektů a jejich vlastností. Téměř veškerá data, se kterými se setkáme, jsou objekty.

Třídy jsou definovány pomocí klíčového slova class, které je následováno jménem třídy a dvojtečkou.

```
>>> class MyClass:
>>>>>> pass
```

Zdrojový kód 21 Syntaxe třídy

Ukázka (Zdrojový kód 21) zobrazuje vytvoření prázdné třídy s názvem MyClass.

1.10.1. Konstruktor __init__()

Konstruktor __init__() je speciální metoda v Pythonu, která je volána při vytváření nové instance třídy. Tato metoda slouží k inicializaci atributů v nově vytvořené instanci třídy.

```
>>> class MyClass:
>>>>>> def __init__(self, parametr1, parametr2):
>>>>>>>>> self.atribut1 = parametr1
>>>>>>>>> self.atribut2 = parametr2
```

Zdrojový kód 22 Syntaxe konstruktoru __init__()

1.10.2. Atributy třídy

Atributy jsou proměnné, které přináleží ke každé instanci třídy. Dělíme je do tří skupin. Datové atributy definují vlastnosti daného objektu, funkční atributy označované jako metody (1.10.4.) definují jeho schopnosti a typové atributy definují pomocné datové typy, které se při práci s danými objekty mohou hodit.

Všechny atributy neboli vlastnosti třídy jsou veřejné. V případě, že chceme učinit atribut privátní, je třeba k němu přidat prefix __. Tento způsob je pouze konvence a interpret Pythonu nebrání v přístupu k atributu mimo třídu.

Atributy se definují stejně jako proměnné uvnitř definice třidy a jsou přístupné pomocí zápisu self.atribut.

```
>>> class Zvire:
>>>>>> def __init__(self, jmeno, druh):
>>>>>>>>> self.jmeno = jmeno
>>>>>>>>> self.druh = druh
```

Zdrojový kód 23 Syntaxe vytvoření atributů

1.10.3. Instance třídy

Instance třídy v Pythonu jsou konkrétní objekty, které jsou vytvořeny na základě definice třídy. Tyto instance mají vlastní atributy a mohou mít různé hodnoty pro tyto atributy. Třída, která danou instanci vytvořila, se nazývá mateřská třída. Instance, vytvořená danou třídou, je vlastní instance.

```
>>> class Zvire:
>>>>>> def __init__(self, jmeno, druh):
>>>>>>>>> self. Jmeno = jmeno
>>>>>>>>> self.druh = druh


>>> zvire1 = Zvire("Micka", "Kočka")
>>> zvire2 = Zvire ("Čip","Pes")
```

Zdrojový kód 24 Vytvoření instance třídy

1.10.4. Metody

Metody jsou v Pythonu funkce, které jsou definované uvnitř třídy a jsou volány pomocí instance této třídy. Metody mohou mít přístup k vlastnostem a metodám třídy, ke které patří. Jsou definovány podobně jako funkce s tím rozdílem, že povinným parametrem metody je první parametr self (1.10.5.).

```
>>> def jmenoMetody(self, parametr1, parametr2):
>>>>>> telo_metody
```

Zdrojový kód 25 Metoda

1.10.5. Parametr self

Parametr self umožňuje metodě přístup ke konkrétnímu objektu. Reprezentuje instanci třídy, na které se metoda volá.

```
>>> class Zvire:
>>>>>> def __init__(self, jmeno, druh):
>>>>>>>>> self. Jmeno = jmeno
>>>>>>>>> self.druh = druh

>>>>>> def vypisDruh(self):
>>>>>>>>> print(f"{self.druh}")

>>> zvire1 = Zvire("Micka", "Kočka")
>>> zvire2 = Zvire ("Čip","Pes")

>>> zvire1.vypisDruh()
Kočka
>>> zvire2.vypisDruh()
Pes
```

Zdrojový kód 26 Parametr self

1.11. Moduly

Moduly v Pythonu umožňují seskupování funkcí do logických celků, které mohou být použity v libovolném počtu programů. Moduly v Pythonu jsou soubory s příponou .py. Mohou být použity pro importování do jiných skriptů.

Python obsahuje několik standardních knihoven, jako jsou math, které obsahují matematické funkce, a konstanty, jako jsou sin, cos a pi. Dále metody random, pro generování náhodných čísel a datetime pro práci s daty a časy. Posledními zmíněnými metodami jsou os pro interakci s operačním systémem a re pro práci s regulárními výrazy.

1.11.1. Vlastní moduly

Moduly jsou soubory s příponou .py, a proto je možné je vytvářet. Název modulu je stejný jako název souboru bez koncovky .py.

```
# zvire.py

>>> class Zvire:
>>>>>> def __init__(self, jmeno, druh):
>>>>>>>>> self. Jmeno = jmeno
>>>>>>>>> self.druh = druh

>>> def vypisDruh(self):
>>>>>>>>> print(f"{self.druh}")
```

Zdrojový kód 27 Vytvoření modulu

1.11.2. Importování modulů

Moduly můžeme importovat několika způsoby. Standardním způsobem, kde importujeme celý modul, je použití příkazu import následovaného názvem modulu. Tento způsob je nejjednodušší a také nejbezpečnější, protože nehrozí možnost kolize názvů.

```
>>> import nazev_modulu
```

Zdrojový kód 28 Importování celého modulu

Dalším způsobem je importování konkrétního objektu z modulu, což nám umožňuje použití objektu z modulu bez zadání prefixu. Pomocí klíčového slova from se určí název modulu a následně se specifikuje požadovaný objekt příkazem import. Je též možné naimportovat více objektů z modulu pouhým přidáním druhého názvu a oddělením čárkou. Tato metoda může způsobit konflikty, protože činí importované objekty přímo přístupné.

```
>>> from nazev_modulu import objekt
```

Zdrojový kód 29 Importování konkrétního objektu z modulu

Posledním z možných způsobů importování je importování všech veřejných objektů z modulu, který umožňuje importovat všechny objekty z modulu do aktuálního jmenného prostoru, což znamená, že lze volat objekty přímo bez prefixu s názvem modulu. Tento způsob se zapisuje podobně jako předešlý s tím rozdílem, že je nahrazen název metody hvězdičkou, která zastupuje všechny metody v modulu.

```
>>> from nazve_modulu import *
```

Zdrojový kód 30 Importování všech veřejných objektů z modulu

Python také umožňuje dát danému modulu vlastní název pomocí přidání as následovaného vlastním názvem modulu.

```
>>> import nazev_modulu as novy_nazev_modulu
```

Zdrojový kód 31 Vlastní pojmenování modulu

1.11.3. Použití modulů

V případě, že máme balíčky importovány, můžeme je začít používat. Používání balíčku se liší podle způsobu importování.

Při využití importování celého modulu můžeme k objektům uvnitř přistupovat pomocí operátoru tečka. Příklad je uveden v následující ukázce.

```
>>> import nazev_modulu
>>> nazev_modulu.potrebniObject()
```

Zdrojový kód 32 Použití modulu pomocí importování celého modulu

Po importování konkrétní funkce z modulu můžeme přímo zavolat požadovaný objekt z modulu bez kvalifikace.

```
>>> from nazev_modulu import objekt
>>> objekt()
```

Zdrojový kód 33 Použití objektu po importování konkrétní funkce z modulu

Použitím importování všech veřejných objektů můžeme objekty používat stejným způsobem jako u předešlého způsobu s tím rozdílem, že v tomto případě můžeme využívat veškeré veřejné objekty daného modulu.

```
>>> from nazev_modulu import *
>>> importovanyObjekt()
```

Zdrojový kód 34 Použití objektu při importování všech veřejných objektů z modulu

1.11.4. Balíčky

Balíček je složka, která obsahuje skupinu modulů a soubor nazvaný __init__.py. Příkladem balíčku v Pythonu může být složka my_package, která obsahuje moduly "module1.py", "module2.py" a "__init__.py". Soubor "__init__.py" může obsahovat definice funkcí, tříd a proměnných, které jsou dostupné všem modulům v balíčku.

2. Maya

Autodesk Maya je software pro vytváření animací, filmů, her a dalších 3D projektů. Maya je jedním z nejpopulárnějších 3D softwarů. Je oblíbený především pro svou flexibilitu a širokou škálu nástrojů.

Maya je postavena na otevřené architektuře a její operace lze skriptovat nebo programovat prostřednictvím dobře dokumentovaného a komplexního aplikačního programovacího rozhraní (API) nebo přímo v jednom z vestavěných skriptovacích jazyků, jako je Maya Embedded Language nebo Python.

Aktuální verze Maya, v době psaní této práce, je verze 2023, která se liší především přepracováním nástroje Boolean Tool, který slouží k vytváření objektů pomocí logických operací, a přidáním nástroje Blue Pencil Tool, pomocí kterého můžeme rychleji kreslit křivky. Dále je zde také vylepšené uživatelské prostředí o Application Home centrum a jsou přidány funkce vyhledávání a zajímavější tutoriály.

3. API Maya

Maya devkit obsahuje C++, Python a .NET API. C++ API poskytuje nejnižší úroveň přístupu k funkčnosti aplikace Maya, což umožňuje vývojářům vytvářet vlastní funkce a nástroje s vysokým výkonem a přesností. Python API poskytuje vývojářům vysokou úroveň abstrakce a rychlost vývoje, což umožňuje snadné prototypování a rychlou iteraci při vývoji nových funkcí a nástrojů. .NET API poskytuje vývojářům možnost vytvářet aplikace pro použití v aplikaci Maya, které využívají výhod .NET Frameworku. To umožňuje vývojářům využít funkce a knihovny .NET pro tvorbu aplikací, které mohou komunikovat s aplikací Maya.

V devkitu jsou zahrnuty dvě verze Python API, Python API 1.0 a Python API 2.0. Verze Python API 1.0 je převedením C++ API do jazyka Python, zatímco verze Python API 2.0 je nově vytvořenou verzí Python API, která je lépe přizpůsobená specifickým vlastnostem jazyka Python a nabízí funkce specifické pro tento jazyk.

API je platformě neutrální a neobsahuje žádné volání specifické pro danou platformu. Zdrojový kód pro dané platformy musí být překompilován. [7]

3.1. Příkazový řádek

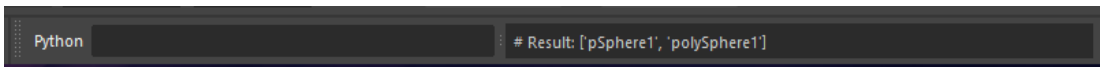
Příkazový řádek slouží ke psaní jednořádkových kódů. Používá se především pro testování před přidáním daného kódu do funkčního kódu nebo pro získání více informací o aktuální scéně.

Příkazový řádek se nachází na spodní části Maya GUI. Lokaci můžeme vidět na obrázku. Na levé straně příkazového řádku můžeme nastavit Python jako skriptovací jazyk.

Obrázek 1 Maya příkazový řádek

Pro příklad můžeme v příkazovém řádku vytvořit jednoduchý objekt. Vložením příkazu cmds.polySphere(r=5) vytvoříme kouli s poloměrem 5 jednotek neboli

centimetrů. V okně vedle příkazového řádku můžeme vidět výsledek našeho příkazu. V tomto konkrétním případě bude výsledek # Result: ['pSphere1', 'polySphere1'].



Obrázek 2 Výsledek příkazového řádku

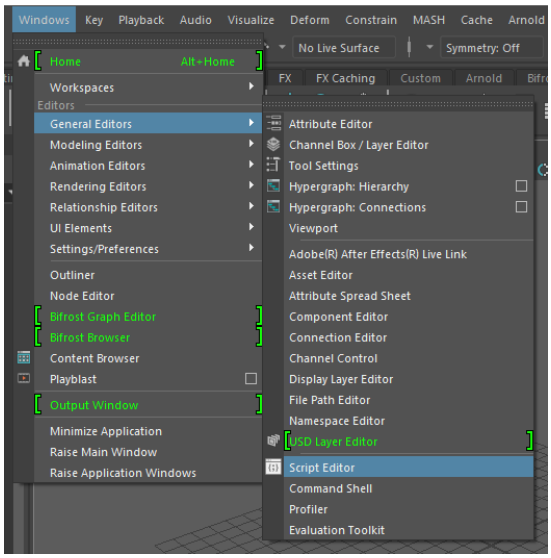
3.2. Skriptovací editor

Hlavním nástrojem pro psaní skriptů v Maye slouží skriptovací editor neboli Script Editor. Editor skriptů je rozhraní pro vytváření krátkých skriptů pro interakci s Mayou. [8]

Obrázek 3 Ikona pro spuštění Skriptovacího editoru

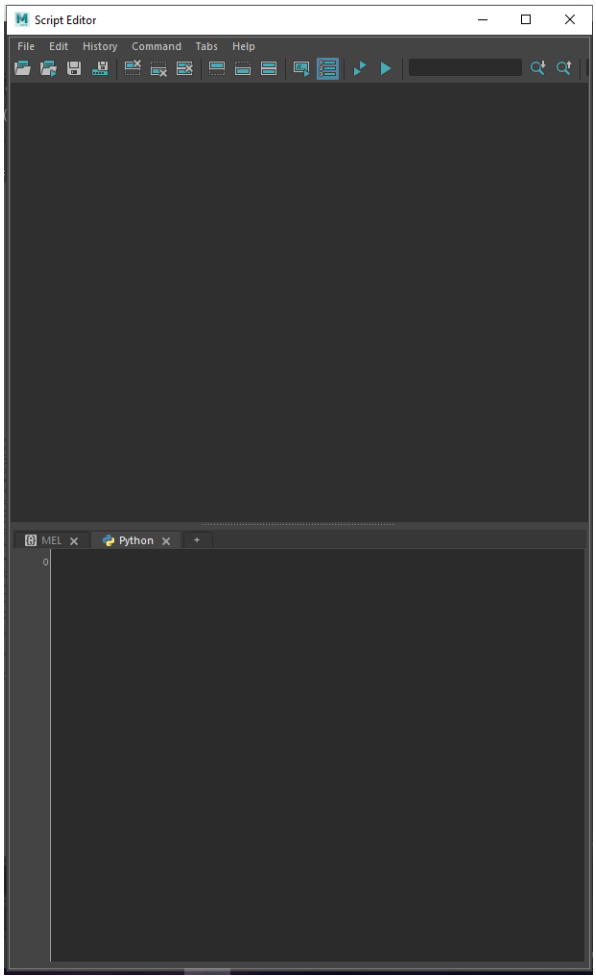
Tento nástroj zobrazíme pomocí ikony v pravém dolním rohu.

Skriptovací panel můžeme také otevřít pomocí okna Window, kde klikneme na General Editors a následně na Script Editor.



Obrázek 4 Druhý způsob spuštění skriptovacího editoru

Samotný skriptovací editor se skládá ze dvou oken. Horní panel obsahuje historii, spodní panel je vstupní panel. Pomocí dvou tlačítek nacházejících se nad vstupním panelem je možné podobně jako v případě příkazového řádku přepnout používaný skriptovací jazyk. V horním panelu okna se nacházejí tlačítka pro ovládání. Kód, zadaný do vstupního panelu, můžeme spustit pomocí dvou tlačítek. První ikona dvou šipek spustí celý skript. Ikona jedné šipky spustí pouze zvýrazněný kód. Pokud není nic zvýrazněno, spustí se automaticky celý blok.



Obrázek 5 Skriptovací editor

3.3. Tvůrčí, dotazovací a upravovací příznaky

Příznaky v Maya se dělí do tří základních skupin. Těmi jsou tvůrčí příznaky (Create flags), dotazovací příznaky (Query flags) a upravovcí příznaky (Edit flags). Tyto nástroje používáme pro tvorbu, dotazování a úpravu prvků scény. S těmito pojmy se setkáme především v dokumentaci Maya, kde jsou použity u jednotlivých skriptů.

Tvůrčí příznaky slouží k vytváření nových prvků, jako jsou objekty, světla, kamery nebo materiály. Tyto příkazy jsou využívány pro vytváření nových objektů na základě specifikovaných parametrů.

Dotazovací příznaky slouží k získávání informací o již existujících prvcích ve scéně. Tímto způsobem může uživatel získat informace o pozici, velikosti, názvu, typu a dalších vlastnostech prvků ve scéně.

Upravovací příznaky slouží k úpravám již existujících prvků ve scéně. Tyto příznaky mohou být použity pro změnu vlastností prvku, jimiž jsou pozice, rotace, velikost nebo vlastnosti materiálu.

Jednotlivé vlastnosti příznaků jsou označovány písmeny „C“, „E“, „Q“ a „M“. Tyto značky můžeme vidět ve sloupci vlastnosti. Zkratka „C“ značí tvůrčí příznaky, „E“ upravovací příznaky a “Q“ je pro dotazovací příznaky. Poslední zkratka „M“ značí, že příznak může mít více argumentů, předávaných buď jako n-tice nebo seznam.

3.4. Importování vestavěných funkcí Pythonu

Chceme-li začít pracovat se skriptováním v Maya, je třeba naimportovat knihovnu maya.cmds. Pomocí této knihovny budeme následně přistupovat k funkcím Maya.

```
>>> import maya.cmds
```

Zdrojový kód 35 Importování knihovny maya.cmds

Tuto naimportovanou knihovnu můžeme následně použít například pro vytvoření kvádru. Ten vytvoříme pomocí příkazu v příkladu (Zdrojový kód 36).

```
>>> import maya.cmds as cmds
>>> cmds.polyCube(sx=10, sy=15, sz=5, h=20)
```

Zdrojový kód 36 Ukázka vytvoření kvádru

Knihovna maya.cmds nenabízí pouze možnost vytvoření objektu, ale poskytuje i spoustu dalších možností. Její možnosti si můžeme zobrazit pomocí příkazu dir(), který nám zobrazí obsah daného objektu.

```
>>> commandList = dir(cmds)
>>> for command in commandList:
>>>>>> print(command)
```

Zdrojový kód 37 Vypsání obsahu objektu maya.cmds

3.5. Tvorba objektů

Ve skriptovacím editoru Maya můžeme vytvářet mnoho druhů objektů, jako jsou například geometrické objekty, světla, kamery a úpravy.

Vytvoření objektu je poměrně jednoduché. V případě, že potřebujeme vytvořit například kouli, stačí zavolat příkaz cmds.polySPhere(). V případě, že nezadáme žádný parametr, automaticky se vytvoří koule s poloměrem 1 jednotky neboli 1 centimetru. Pokud chceme nastavit vlastní rozměry, je třeba zadat parametr r.

```
>>> import maya.cmds as cmds


>>> cmds.polySphere(r=2)
```

Zdrojový kód 38 Vytvoření objektu

3.6. Mazání objektů

Pro vymazání objektu můžeme použít příkaz cmds.delete(). Tento příkaz bez zadaných parametrů smaže všechny vybrané objekty. Pokud chceme vymazat objekty bez vybrání, stačí jako parametr zadat jejich název. Chceme-li tedy vymazat objekt koule1, bude příkaz vypadat následovně: cmds.delete (‚koule1‘). Pro vymazání všech objektů ve scéně je potřeba zadat cmds.delete(all=True). V následující ukázce jsou zobrazeny všechny zmíněné způsoby mazání.

```
>>> import maya.cmds as cmds
```

```
>>> cmds. delete()
>>> cmds.delete(‚koule1‘)
>>> cmds.delete(all=True)
```

Zdrojový kód 39 Mazání objektů

3.7. Tvorba dialogového okna

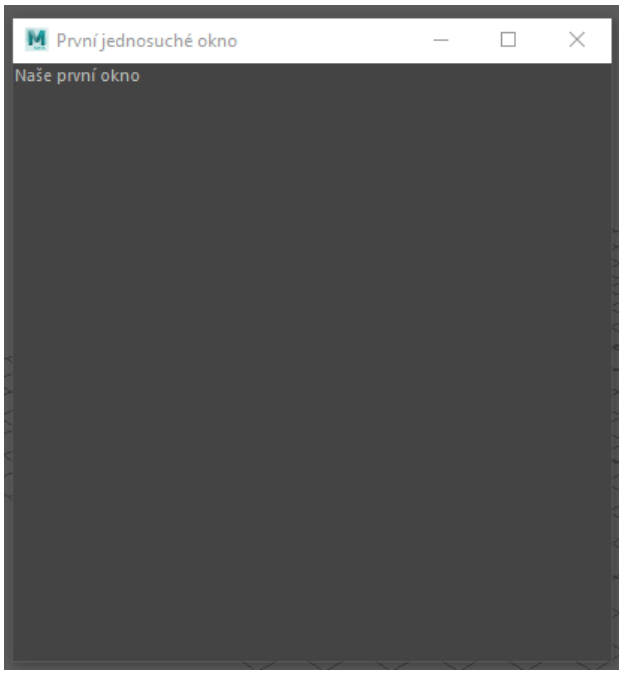
Dialogová okna v aplikaci Maya jsou užitečná pro interakci uživatele s nástroji a skripty v prostředí Maya. Mohou být použita k mnoha účelům, např. k zadávání vstupů od uživatele, zobrazení výsledků skriptů, výběru položek z menu a mnoha dalším. Další možností je použití dialogových oken k zobrazování výsledků skriptů a interakci s uživatelem. Dialogová okna mohou být také použita k výběru různých položek z menu nebo k zobrazení pokročilých možností nastavení.

Dialogové okno můžeme jednoduše vytvořit pomocí příkazu cmds.window(). Pomocí parametru title mu můžeme přiřadit titulek a parametr widthHeight, nám umožní nastavit rozměry. Pomocí metody cmds.columnLayout() nastavíme rozvržení okna a pomocí metody cmds.text() vložíme do dialogu požadovaný text. V případě, že bychom nenastavili rozložení okna, nebude moci Maya správně rozložit vložené prvky a výsledek bude chybový. Takto nastavené okno zobrazíme pomocí metody cmds.showWindow(), která bude mít jako parametr námi vytvořené okno. Pro lepší použitelnost kódu si okno vytvoříme ve funkci. Vytvoření jednoduchého okna je zobrazeno v následujícím příkladu (Zdrojový kód 40).

```
>>> import maya.cmds as cmds


>>> def showUI():
>>>>>> myWin = cmds.window(title="První jednoduché okno",
widthHeight=(400, 400))
>>>>>> cmds.columnLayout()
>>>>>> cmds.text(label="Naše první okno")
>>>>>> cmds.showWindow(myWin)
>>> showUI()
```

Zdrojový kód 40 Vytvoření dialogového okna

Výsledek tohoto kódu můžeme vidět na následujícím obrázku.



Obrázek 6 Dialogové okno

3.7.1. Tvorba tlačítka

Tlačítka jsou důležitými součástmi dialogových oken. Můžeme je použít pro spuštění různých akcí, například potvrzení, zrušení nebo zavření dialogového okna.

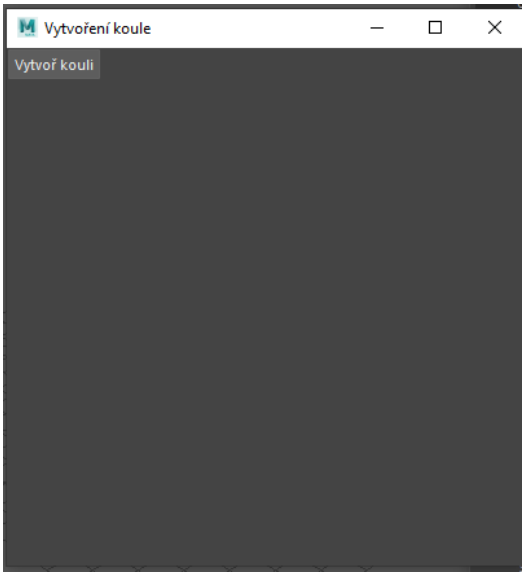
Tlačítko do dialogového okna můžeme přidat pomocí příkazu cmds.button(). Pomocí parametru lable zadáme tlačítku titulek a pomocí parametru command mu přidělíme požadovanou metodu. V následujícím zdrojovém kódu je zobrazeno jednoduché vytvoření dialogového okna s tlačítkem, které vytvoří kouli.

```
>>> import maya.cmds as cmds

>>> def createSphere (args):
>>>>>> cube = cmds.polySphere(name="koule")

>>> def showUI():
>>>>>> myWin = cmds.window(title="Vytvoření koule", widthHeight=(400,
400))
>>>>>> cmds.columnLayout()
>>>>>> cmds.button(label="Vytvoř kouli", command=createSphere)
```

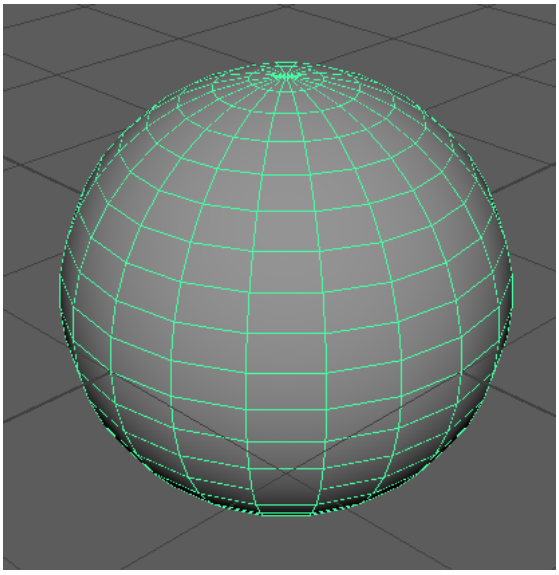
```
>>>>>> cmds.showWindow(myWin)
>>> showUI()
```

Zdrojový kód 41 Přidání tlačítka do dialogového okna

Pomocí tohoto příkazu se nám vytvoří okno zobrazené na obrázku 7. Obrázek 8 zobrazuje výsledek po kliknutí na námi vytvořené tlačítko.



Obrázek 7 Dialogové okno s tlačítkem



Obrázek 8 Výsledek dialogového okna s tlačítkem

3.7.2. Tvorba vstupních ovládacích prvků

Další důležitou součástí v dialogových oknech jsou vstupní ovládací prvky. Tyto prvky nám umožňují zadávat různé informace a nastavení při používání dialogových oken.

Mezi nejpoužívanější vstupní ovládací prvky patří například textové pole, rozbalovací seznam nebo posuvník.

Jednoduché textové pole můžeme vytvořit pomocí příkazu cmds.intField(). Tento příkaz může mít parametr minValue, nebo maxValue, který nám umožní určení minimální a maximální hodnoty.

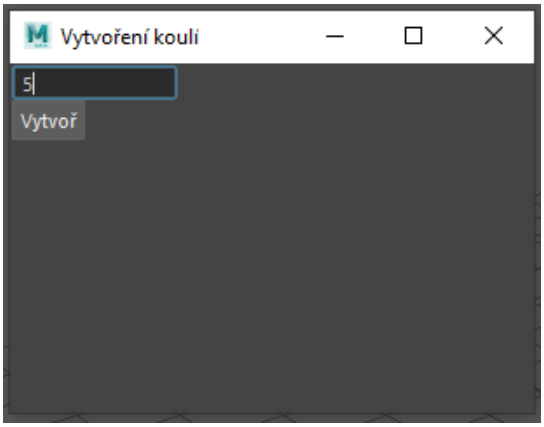
V následujícím zdrojovém kódu můžeme vidět kód, který vytvoří dialogové okno s jedním tlačítkem a jedním textovým polem. Hodnotu zadanou do textového

pole uložíme do globální proměnné vstupPocetKouli. Po stisknutí tlačítka s názvem Vytvoř se spustí metoda s názvem vytvorKoule. Tato metoda vytváří koule na základě hodnoty vložené v textovém poli. Počet koulí, které se mají vytvořit, se určuje pomocí proměnné pocetKouli, do které se uloží vstup z textového pole, který je převeden na celé číslo. K vytvoření koulí se použije cyklus, jehož počet opakování je roven hodnotě v proměnné pocetKouli. Koule jsou vytvořeny s rozestupem 2.

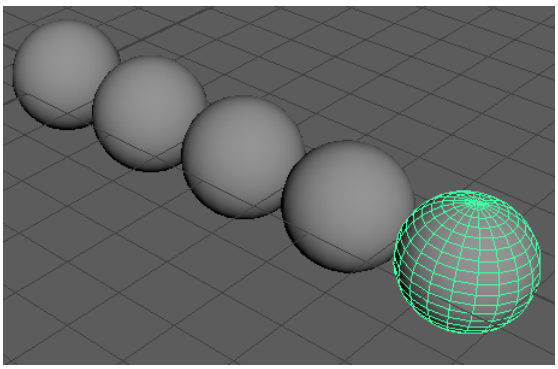
```
>>> import maya.cmds as cmds
>>>
>>> global vstupPocetKouli

>>> def showUI():
>>>>>> global vstupPocetKouli

>>>>>> myWin = cmds.window(title="Vytvoření koulí", widthHeight=(300,
200))
>>>>>> cmds.columnLayout()
>>>>>> vstupPocetKouli = cmds.intField(minValue=1)
>>>>>> cmds.button(label="Vytvoř", command=vytvorKoule)
>>>>>> cmds.showWindow(myWin)

>>> def vytvorKoule(args):
>>>>>> global vstupPocetKouli
>>>>>> pocetKouli = cmds.intField(vstupPocetKouli, query=True,
>>>>>> value=True)
>>>>>> for i in range(pocetKouli):
>>>>>>>>> cmds.polySphere()
>>>>>>>>> cmds.move((i * 2), 0, 0)

>>>showUI()
```

Zdrojový kód 42 Dialogové okno s tlačítkem a textovým vstupem

Na následujících obrázcích můžete vidět výsledné dialogové okno a vytvořené objekty na základě zadaných dat.



Obrázek 9 Dialogové okno s tlačítkem a textovým polem



Obrázek 10 Výsledek dialogového okna s textovým polem

3.7.3. Vytvoření více karet

Vytváření více karet v dialogovém okně může být užitečné pro organizaci různých funkcionalit do logických skupin a snadné přepínání mezi nimi pomocí karet. Toto uspořádání umožňuje uživatelům rychleji najít to, co potřebují, a zjednodušit tak ovládání rozhraní.

Při vytváření okna organizovaného do karet musíme nejdříve nastavit rozvržení na karty pomocí příkazu cmds.tabLayout(), které si uložíme do proměnné tabs. První kartu vytvoříme pomocí příkazu cmds.columnLayout(), který si uložíme do proměnné prvniKarta. Do proměnné prvniKarta se nám uloží název nové karty. Dále je třeba nastavit jméno karty a rozložení karet pomocí příkazu cmds.tabLayout(self.tabs, edit=True, tabLabel=[prvniKarta , 'Koule']). Self.tabs nám nasměruje kartu na rozložení karet a pomocí tabLabel nastavíme na kartu prvniKarta název Koule.

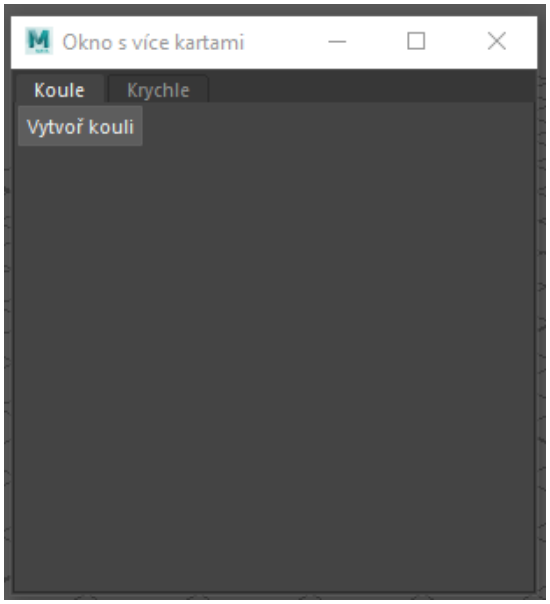
V případě, že chceme přidávat další karty, je třeba přepnout se zpět do rozložení karet pomocí příkazu cmds.setParent("..").

V následující ukázce zdrojového kódu je zobrazeno vytvoření dvou karet, kde každá karta obsahuje jedno tlačítko pro vytvoření objektu. V prvním případě je objektem koule a v druhém krychle. Každý nově vytvořený objekt se nám posune díky proměnné cislo, která uchovává počet vytvořených objektů.

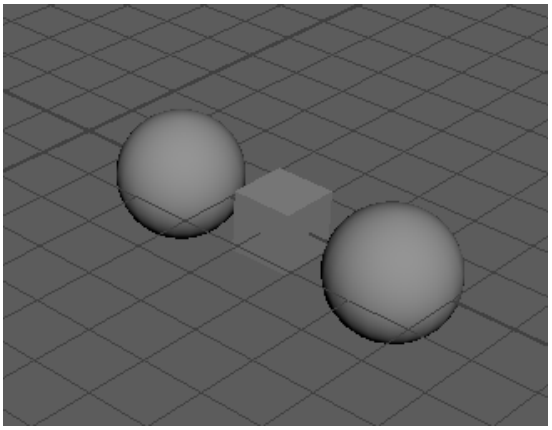
```
>>> import maya.cmds as cmds

>>>class TabExample:
>>>>>> cislo = 0

>>>>>> def __init__(self):
>>>>>>>>> self.win = cmds.window(title="Okno s více kartami",
widthHeight=(300, 300))
>>>>>>>>> self.tabs = cmds.tabLayout()

>>>>>>>>> prvniKarta = cmds.columnLayout()
>>>>>>>>> cmds.tabLayout(self.tabs, edit=True, tabLabel=[prvniKarta , 'Koule'])
>>>>>>>>> cmds.button(label="Vytvoř kouli", command=self.createSphere)
>>>>>>>>> cmds.setParent("..")

>>>>>>>>> druhaKarta = cmds.scrollLayout()
>>>>>>>>> cmds.tabLayout(self.tabs, edit=True, tabLabel=[druhaKarta,
'Krychle'])
>>>>>>>>> cmds.button(label="Vytvoř krychli", command=self.createCube)
>>>>>>>>> cmds.columnLayout()

>>>>>>>>> cmds.showWindow(self.win)

>>>>>> def createSphere(self, args):
>>>>>>>>> self.cislo = self.cislo + 1
>>>>>>>>> cmds.polySphere()
>>>>>>>>> cmds.move((self.cislo * 2), 0, 0)

>>>>>> def createCube (self, args):
>>>>>>>>> self.cislo = self.cislo + 1
>>>>>>>>> cmds.polyCube()
>>>>>>>>> cmds.move((self.cislo * 2), 0, 0)
```

```
>>> TabExample()
```

Zdrojový kód 43 Vytvoření dialogového okna s více kartami



Obrázek 11 Dialogové okno s více kartami



Obrázek 12 Výsledek dialogového okna s více kartami

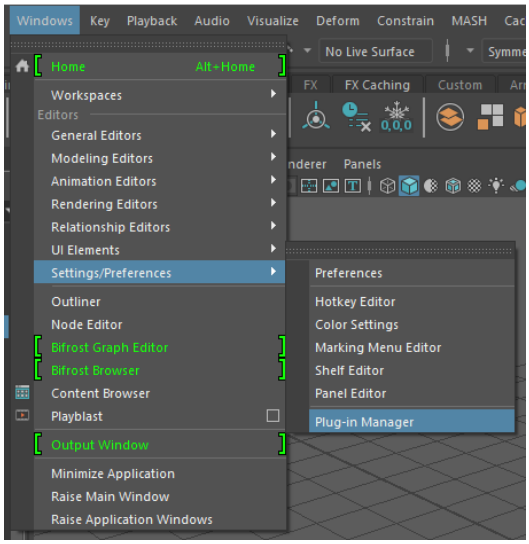
Obrázek 11 znázorňuje okno vytvořené předchozím zdrojovým kódem, které obsahuje dvě karty. Obrázek 12 zobrazuje objekty, které byly vytvořeny v následujícím pořadí: nejprve bylo kliknuto na tlačítko v kartě Koule, poté na tlačítko v kartě Krychle a nakonec bylo tlačítko pro vytvoření koule stisknuto znovu.

3.8. Zásuvné moduly

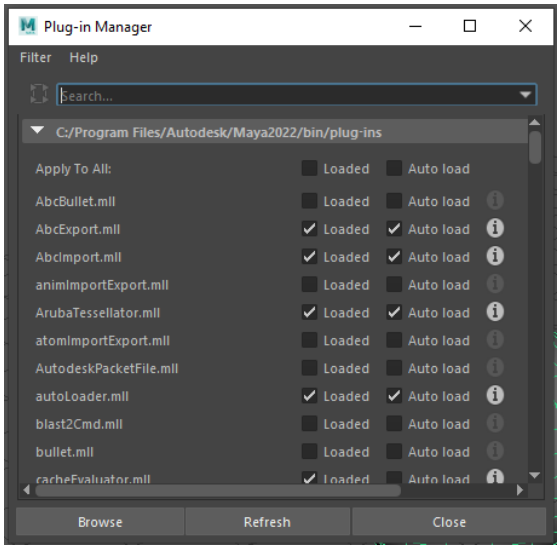
Zásuvné moduly jsou externí knihovny, které Maya Plug-in Managere načítá z C:\Users\<username>\Documents\maya\<version>\plug-ins adresáře. Můžeme je používat k definování vlastních příkazů, uzlů, omezení, překladačů souborů a deformátorů. Tvorba modulů je popsána v kapitole vlastní moduly (1.11.1.).

3.8.1. Načítání zásuvných modulů

Pro načítání zásuvných modulů je nejprve nutné načíst zásuvné moduly do Plug-In Manageru. Ten nalezneme na cestě Windows → Settings/Preferences → Plug-in Manager. Pomocí tlačítka Browse můžeme přidat další moduly. Po načtení se modul automaticky načte do cmds modulu.



Obrázek 13 Cesta k Plug-in Manageru



Obrázek 14 Plug-in Manager

4. Praktická část

V rámci praktické části byl vytvořen webový kurz, který zájemce provede základy Pythonu a skriptování v Autodesku Maya. Kurz je rozdělen do deseti lekcí, kde prvních pět lekcí popisuje jazyk Python a zbylých pět lekcí je věnováno Autodesku Maya. Kurz provede zájemce základy jazyka Python, naučí uživatele základní orientaci v prostředí Maya a nakonec jej naučí základní tvorbu objektů a dialogových oken.

Kurz reprodukuje text obsažený v teoretické části této práce.

Tento kurz je vytvořen pomocí technologií HTML 5, CSS a Javascript. HTML je značkovací jazyk používaný pro tvorbu webových stránek. V této práci tvoří základ webové stránky.

CSS je jazyk používaný pro popis vzhledu a formátování webových stránek a v této práci tvoří hlavní vizuální prvek webové stránky. Činí také webový kurz responzibilní.

Javascript je skriptovací jazyk, který se používá na webových stránkách pro interaktivní prvky, jako jsou například animace, formulářové prvky a dynamické aktualizace obsahu. V této práci ovládá animační prvky webového kurzu, především přepínání textů.

4.1. Grafické prostředí webového kurzu

Celý kurz je stavěn jako uživatelsky přívětivý. Pro tvorbu byly zvoleny barvy, které jsou obsaženy v logu Python a Maya a jsou rozšířené o oranžovou barvu pro zvýraznění důležitých prvků. Jednotlivé bloky Python a Maya jsou rozlišené modrou a žlutou barvou. Veškerá tlačítka jsou animována na detekci kurzoru.

4.1.1. Úvodní strana

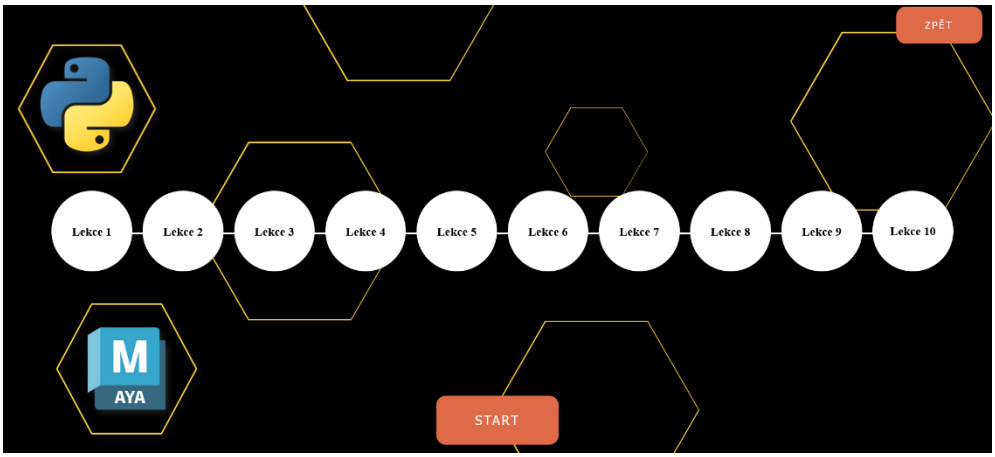
Úvodní strana zobrazuje stručné informace o kurzu a jeho cílech, které jsou znázorněny i graficky pomocí obrázků. Hlavní tlačítko PLAY umožňuje spuštění kurzu. Celá úvodní strana je navržena přehledně a jednoduše, aby bylo vše jasně viditelné na první pohled.

Obrázek 15 Grafické prostředí úvodní strana

Obrázek 16 Mobilní verze úvodní strana

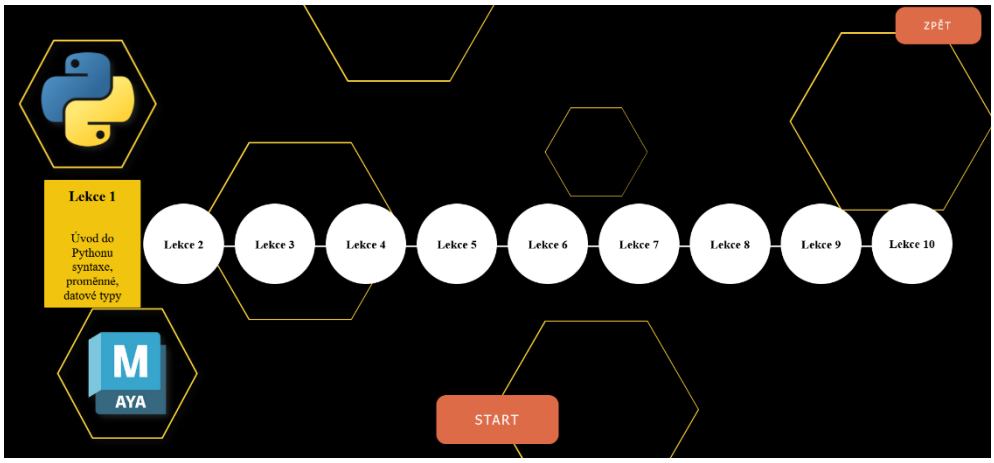
4.1.2. Osnova kurzu

Po zahájení kurzu se uživateli zobrazí osnova kurzu. Tato stránka obsahuje osnovu, tlačítko START pro zahájení kurzu a tlačítko ZPĚT na ukončení kurzu.



Obrázek 17 Grafické prostředí osnova

Na první pohled je v osnově viditelný pouze seznam lekcí. To se ale změní po najetí myší na vybraný kurz. V tom případě se uživateli zobrazí nadpis kapitoly. Uživatel může spustit pouze konkrétní lekci pomocí jednotlivých lekcí v osnově.



Obrázek 18 Osnova rozšířené informace

Obrázek 19 Mobilní verze osnova

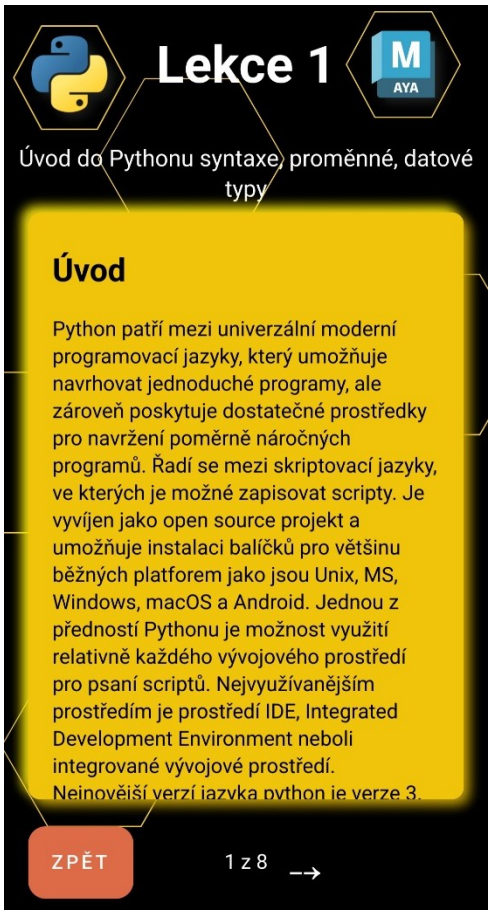
V případě mobilní verze jsou názvy lekcí přímo viditelné. Hlavním důvodem této změny je snaha, aby byl kurz intuitivní a přehledný.

4.1.3. Jednotlivé lekce

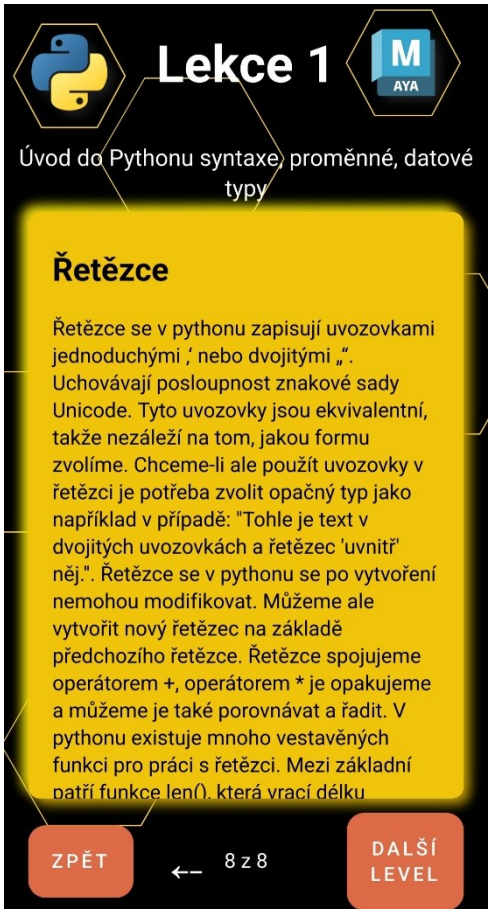
Pokud se uživatel rozhodne spustit kurz, zobrazí se uživateli stránka obsahující hlavní blok s textem kurzu a čtyři tlačítka. Hlavní tlačítka jsou na přepínání textových bloků v rámci levelu. Při prvotním spuštění se uživateli zobrazí pouze tlačítko na posunutí vpřed, po stisknutí tohoto tlačítka se uživateli zobrazí tlačítko na vrácení na předchozí text. Další viditelné tlačítko je tlačítko ZPĚT pro návrat do osnovy. Poslední tlačítko se zobrazí až po dokončení všech textových bloků a slouží pro postup do následujícího levelu. V případě, že je text delší, než je velikost textového pole, je na pravé straně zobrazen scrollBar (posuvník), který umožňuje zobrazení textu, který se do pole již nevejde.

Obrázek 20 Grafické rozhraní začátek levelu

Obrázek 21 Grafické rozhraní konec levelu



Obrázek 22 Mobilní verze začátek levelu



Obrázek 23 Mobilní verze konec levelu

4.1.4. Dokončení kurzu

Po dokončení kurzu se uživateli zobrazí poslední okno obsahující oznámení o dokončení kurzu a tlačítko, které uživatele vrátí zpět na úvodní stránku.

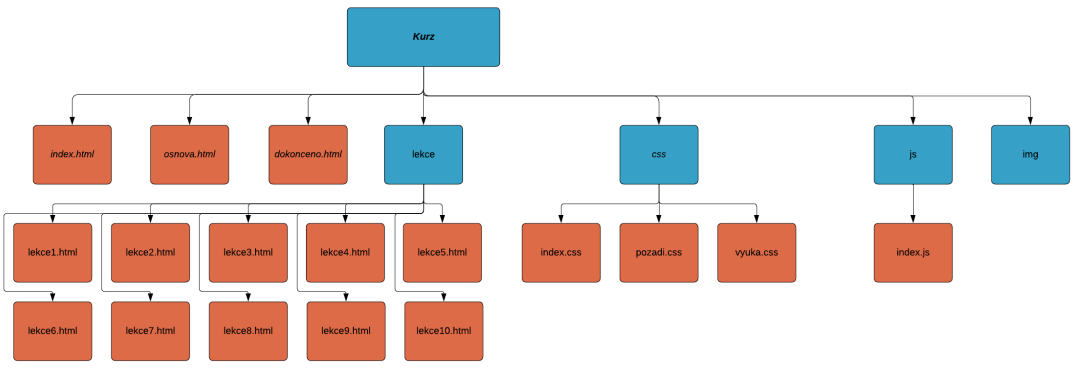
Obrázek 24 Grafické rozhraní kurz dokončen

Obrázek 25 Mobilní verze kurz dokončen

4.2. Programová část

Soubory kurzu jsou rozloženy podle diagramu na následujícím obrázku. Modré elementy znázorňují složky a oranžové soubory.

Jednotlivé lekce jsou rozděleny do jednotlivých html souborů, a to z důvodu přehlednosti.



Obrázek 26 Diagram souborů

Následující obrázek obsahuje kód souboru index.html.

```html
<!DOCTYPE html>
<html lang="cz">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Skriptování v programu Autodesk Maya</title>
        <link id="styly" rel="stylesheet" href="css/index.css">
        <link id="styly" rel="stylesheet" href="css/pozadi.css">
    </head>
    <body>
        <!-- Pozadí stránky -->
        <div class="pozadi">···
        </div>

        <!-- Logo Pythonu -->
        <div class="logo_python">
            <img onclick="location.href='lekce1.html'" src="/img/python-logo-only-stroke.png" alt="Python logo">
        </div>

        <!-- Logo Maya -->
        <div class="logo_maya">
            <img onclick="location.href='lekce6.html'" src="/img/maya-stoke.png" alt="Maya logo">
        </div>

        <!-- Informační karta -->
        <div class="card">
            <!-- Blok s obrázky -->
            <div class="img">
                <img  class="male" src="/img/dialogove_okno_vice_karet.png"
                    alt="Dialogové okno s více kartami">
                <img  class="male" src="/img/vysledek_vice_oken.png"
                    alt="Výsledek dialogového okna s více kartami">
            </div>

            <!-- Blok s textem -->
            <div class="text">
                <p class="h3">Skriptování v programu Autodesk Maya</p>
                <p class="p">Kurz je zaměřen na seznámení se scriptovacím
                    rozhraním v Autodesc Maya. Jeho cílem je naučit
                    scriptovat
                    pomocí Pythonu. Kurz je rozdělen na 10 kekcí, kde první
                    polovina je věnována Pythonu a druhá polovina Maye. Cílem
                    kurzu bude vytvořit objekt na obrázku
                    pomocí sckriptování ve vestavěném rozhraní.</p>
                <button class="kurzBtn" onclick="location.href='osnova.html'">
                    Play
                </button>
            </div>
        </div>
    </body>
</html>
```

Obrázek 27 Soubor index.html

CSS prvky jsou rozděleny do tří souborů. První soubor index.css obsahuje styly úvodních stránek jako je index.html, osnova.html a dokonceno.html. Soubor pozadi.css obsahuje prvky upravující pozadí jednotlivých stránek, a to především tvorbu hexagonů. CSS soubory jsou rozděleny do dvou částí, kde první je věnována obrazovkám větším než 600 px, tedy počítačům a tabletům, a druhá část obrazovkám menším než 600 px, tedy mobilům.

```css
/* PC */
@media only screen and (min-width: 600px) {
  body {
    --blue-green: #37a0c6;
    --paynes-gray: #34657b;
    --burnt-sienna: #de6b48;
    --mustard: #ffd53d;
    --jonquil: #f1c40f;
    display: flex;
    height: 100vh;
    align-items: center;
    justify-content: center;
    justify-items: center;
    text-align: center;
    overflow: hidden;
    background-color: black;
  }

  #visual {
    height: auto;
    width: 100vw;
  }

  h1 {
    font-size: 40px;
    color: rgb(255, 255, 255);
  }

  h2 {
    font-size: 30px;
    color: rgb(255, 255, 255);
  }

  .uvod {
    position: relative;
    width: 90vw;
    height: 100vh;
    display: flex;
    flex-direction: row;
    align-items: center;
    justify-content: space-between;
  }
```

Obrázek 28 Ukázka CSS kódu pro počítač

```css
/* Mobil */
@media only screen and (min-width: 600px) {
  body {
    --blue-green: #37a0c6ff;
    --paynes-gray: #34657bff;
    --burnt-sienna: #de6b48ff;
    --mustard: #ffd53dff;
    --jonquil: #f1c40fff;
    margin: 0;
    padding: 0;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    text-align: center;
    background-color: black;
    overflow-x: hidden;
  }

  #visual {
    height: auto;
    width: 100vw;
  }

  h1 {
    font-size: 40px;
    color: rgb(255, 255, 255);
  }

  .uvod {
    position: relative;
    width: 90vw;
    height: 100vh;
    display: flex;
    flex-direction: row;
    align-items: center;
    justify-content: space-between;
  }
```

Obrázek 29 Ukázka CSS kódu pro mobil

Soubor index.js ovládá přepínání textů v jednotlivých kurzech. Na následujícím obrázku je zobrazena část JavaScript kódu ze souboru index.js. První metoda kódu umožňuje zobrazit kartu na určeném indexu. Dále tento kód obsahuje handler, který čeká na stisknutí tlačítka prevButton. Následně po zaznamenání akce na tomto tlačítku zavolá metodu showCard a do parametru n vloží pozici o jedno menší. Další aktivitou, kterou tento handler vykonává, je změna textu, který označuje aktuální pozici. Poslední dvě akce jsou zobrazení tlačítka nextButton, pokud je skryté, a skrytí tlačítka prevButton, pokud jsme na prvním textu.

```
let currentCard = 0;
let cislo = document.getElementById("cislo");
const cards = document.querySelectorAll(".info");
const prevButton = document.querySelector("#prev");
const nextButton = document.querySelector("#next");

prevButton.style.visibility = "hidden";
cislo.innerHTML = currentCard + 1 + " z " + cards.length;

/* Funkce pro zobrazení karty */
function showCard(n) {
  cards[currentCard].classList.remove("active");
  currentCard = (n + cards.length) % cards.length;
  cards[currentCard].classList.add("active");
}

/* Handler čekající na kliknutí na tlačtko předchozí text */
prevButton.addEventListener("click", () => {
  showCard(currentCard - 1);
  cislo.innerHTML = currentCard + 1 + " z " + cards.length;
  /* Zobrazení tlačítka následující pokud je skryté */
  if (nextButton.style.visibility == "hidden") {
    nextButton.style.visibility = "visible";
  }
  /* Skrytí tlačítka předchozí, pokud jsme na prvním textu */
  if (currentCard == 0) {
    prevButton.style.visibility = "hidden";
  }
});
```

Obrázek 30 Příklad JavaScript kódu

5. Závěr

Téma bakalářské práce pro mne bylo velmi přínosné, jelikož jsem měla alespoň okrajově možnost si prakticky vyzkoušet, jak lze pracovat se 3D grafikou. Vzhledem ke vzrůstající potřebě a možnostem tohoto odvětví, bylo vyhledávání a následné uplatnění nabytých informací velmi zajímavé, a to zejména v části skriptování. Jelikož je Python nejdůležitější částí skriptování, tak se mu bakalářská práce věnovala podrobněji. To je důvod, díky kterému je tato práce přínosnější pro osoby, které nemají mnoho zkušeností s jazykem Python, ale zároveň nejsou úplní nováčci v programování či skriptování v jiných jazycích. Pokud by práce měla oslovit i úplné začátečníky ve skriptování v Maye pomocí Pythonu, muselo by se v teoretické části zajít ještě do větších detailů a podrobněji se zaměřit na API Maya a samotné skriptování a modelování.

V rámci praktické části této práce byl vytvořen webový kurz v rozsahu 10 lekcí, který obsahuje články z teoretické části. Tento kurz je určen především pro začátečníky ve skriptování v jazyce Python, kteří mají alespoň základní zkušenosti z jiných programovacích či skriptovacích jazyků. Kurz uživatele seznámí se základními principy jazyka Python a provede je základy skriptování v Maya.

Při zjišťování informací a podkladů ke tvorbě kurzu, věnující se Pythonu, jako skriptovacímu jazyku v Autodesk Maya, nebyl nalezen žádný podobný kurz, ani takové množství informací které by byly takto uceleně interpretovány českému uživateli, a navíc v jeho rodném jazyce. To lze považovat za přednost kurzu samotného, navíc, když jsou textové informace obohaceny obrázky pro lepší názornost a pochopení. Neucelené informace a podklady byly základem pro vznik tohoto kurzu, aby se ostatní uživatelé mohli lépe a snáze seznámit se základy dané problematiky. Jedním z dalších důvodů je, že je Python rozšířený a intuitivní jazyk. Je relativně snadné naučit se ovládat základní znalosti práce s ním, proto použití jazyka Python ve 3D modelování odkrývá nové možnosti pro uživatele, kteří mají zájem rozvíjet své znalosti ve 3D modelování a chtějí rozšířit své znalosti ve skriptování.

Webový kurz je dostupný na adrese https://kurz.s-weby.cz.

6. Použitá literatura

1 PECINOVSKÝ, Rudolf. *Začínáme programovat v jazyku Python*. 2. přepracované a rozšířené vydání. Praha: Grada Publishing, 2022, 316 stran: ilustrace; 24 cm. ISBN 978-80-271-3609-4.

2 Python 3 výukový kurz, 2021. 2. Brno: Computer Press. ISBN 978-80-251-5030-6.

3 PECINOVSKÝ, Rudolf. *Začínáme programovat v jazyku Python*. Praha: Grada Publishing, 2020, 270 stran: ilustrace; 24 cm. ISBN 978-80-271-1237-1.

4 *Python 3.4.10 documentation* [online]. Wilmington, Delaware, USA: Python Software Foundation, 2019 [cit. 2023-02-24]. Dostupné z: https://docs.Python.org/3.4/index.html

5 PYTHON SOFTWARE FOUNDATION. *Data Structures* [online]. 2014 [cit. 2014-10-12]. Dostupný z WWW: https://docs.Python.org/3.4/tutorial/datastructures.html

6 Lekce 6 - Cykly v Pythonu. *ITnetwork* [online]. Praha: David Čápka [cit. 2023-03-08]. Dostupné z: Lekce 6 - Cykly v Pythonu Zdroj: https://www.itnetwork.cz/Python/zaklady/cykly-v-Pythonu

7 The Maya API. *Autodesk MAYA 2022* [online]. San Francisco, California: Autodesk, 2023 [cit. 2023-04-03]. Dostupné z: https://help.autodesk.com/view/MAYAUL/2022/ENU/?guid=Maya_SDK_Maya_API_introduction_API_Basics_html

8 MECHTLEY, Adam a Ryan TROWBRIDGE. *Maya Python for Games and Film*. 1. United States of America: Elsevier, 2012. ISBN 978-0-12-378578-7.

9 Maya Python Plug-in Learning Path. *Autodesk Maya 2022* [online]. USA: Autodesk, 2023 [cit. 2023-04-20]. Dostupné z: https://help.autodesk.com/view/MAYAUL/2022/ENU/?guid=Maya_SDK_Maya_Python_API_Maya_Python_Plug_in_Learning_html