**Vyšší odborná škola a Střední průmyslová škola elektrotechnická Olomouc,**

**Božetěchova3**

**PRAKTICKÁ ZKOUŠKA Z ODBORNÝCH PŘEDMĚTŮ**

Tutoriál na WxPython

2020 Filip Plachý

Prohlašuji, že jsem praktickou zkoušku z odborných předmětů vypracoval samostatně a všechny prameny jsem uvedl v seznamu použité literatury.

……………………………  
 jméno a příjmení žáka

Chtěl bych vyslovit poděkování panu Ing. Markovi Nožkovi za odborné konzultace a poskytnuté informace.

……………………………  
 jméno a příjmení žáka

Prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé práce nebo její části se souhlasem školy.

……………………………  
 jméno a příjmení žáka

# Abstrakt

Cílem diplomové práce je vytvořit sbírku tutoriálů popisující základy knihovny wxWidgets v programovacím jazyce Python. Samotná knihovna pythonu podporující wxWidgets se nazývá wxPython. Jedná se o alternativu k vytváření aplikací s uživatelským grafickým rozhraním.

Sbírka je určena pro studenty programování začínající s GUI, kteří již mají za sebou základní zkušenosti s pythonem. Součástí tutoriálů je úvod do objektově orientovaného programování, které se při tvorbě grafického rozhraní využívá. Po dokončení základních tutoriálů je součástí sbírky i sada názorných příkladů, řešící základní jednoduché aplikace.

# Obsah

[Obsah 5](#_Toc64018250)

[Úvod 6](#_Toc64018251)

[1. Teoretická část 7](#_Toc64018252)

[1.1 Python 7](#_Toc64018253)

[1.1.1 Úvod do jazyka python 7](#_Toc64018254)

[1.1.2 Historie 7](#_Toc64018255)

[1.2 wxPython 10](#_Toc64018256)

[1.2.1 Historie wxPythonu 10](#_Toc64018257)

[1.3 wxWidgets 10](#_Toc64018258)

[1.3.1 Historie wxWidgets 11](#_Toc64018259)

[1.4 Jupyter Notebook 11](#_Toc64018260)

[1.4.1 Historie Jupyter Notebooku 11](#_Toc64018261)

[1.5 Project Jupyter 12](#_Toc64018262)

[1.6 IPython 12](#_Toc64018263)

[1.7 NbViewer 13](#_Toc64018264)

[1.8 GitHub 13](#_Toc64018265)

[1.8.1 Git 14](#_Toc64018266)

[1.9 Objektově orientované programování 14](#_Toc64018267)

[1.9.1 Programovací paradigmata 14](#_Toc64018268)

[1.9.2 OOP 16](#_Toc64018269)

[1.9.3 Funkcionalita OOP v Pythonu 17](#_Toc64018270)

[2. Praktická část 18](#_Toc64018271)

[2.1 Prostředí Pythonu 18](#_Toc64018272)

[Závěr 19](#_Toc64018273)

[Seznam použité literatury 20](#_Toc64018274)

[Seznam obrázků a tabulek 21](#_Toc64018275)

[Přílohy 22](#_Toc64018276)

# Úvod

Práce se rozděluje na 2 části. Teoretická a praktická. Teoretická obsahuje dopodrobna popsané nástroje a funkce, které byly při tvorbě tutoriálu využity. V praktické části si rozebereme samotný tutoriál knihovny wxPython. Tutoriál je určen pro studenty s již základními zkušenosti v programovacím jazyce python. Při práci s wxPython je potřeba mít i znalost Objektově orientovaného programování (OOP), tudíž součástí sady tutoriálů je i lekce do OOP. Cílem práce je vytvořit představu a znalostní základ, se kterým student zvládne vytvářet aplikace s grafickým uživatelským rozhraním.

# Teoretická část

Teoretická část obsahuje základní informace všech programů a nástrojů, které byly při práci využity.

## Python[[1]](#footnote-2)

Cílem této podkapitoly je obecný úvod do programovacího jazyka Python.

### Úvod do jazyka python

Python je univerzální programovací jazyk[[2]](#footnote-3).

Jazyk paří mezi tzv. "interpretované jazyky". To znamená, že napsaný zdrojový kód v jazyce Python je převeden (interpretován) pomocí programu (interpreter/tlumočník) do jazyka, se kterým pracuje počítač. Samotný tlumočník pro Python je k dispozici na všech operačních systémech přímo na stránkách [www.python.org](https://www.python.org).

Pro práci s pythonem slouží tzv. „IDE“ (Integrated Development Environment) , česky: „editor“, který pomáhá uživateli s formátováním kódu tak, aby byl nejen přehledný pro uživatele, ale také aby se dal převést do formátu proveditelného počítačem. Nejznámějšími editory jsou: Pycharm, Pydev, Visual Studio Code, VIM, Atom/Atom-IDE, IDLE, Spyder...

Python byl navržen tak, aby se jeho veškeré funkce nacházely přímo v jádru programovacího jazyka.

Má jednodušší a méně přeplněnou syntaxi a gramatiku, např. díky využití mezer. Další jeho výhodou je rozšiřitelnost o další knihovny/moduly.

### Historie

Programovací jazyk Python byl navrhnut mezi roky 1990-1991 holaňdanem Guidem van Rossumem[[3]](#footnote-4) v národním výzkumném institutu pro matiku a informatiku (CWI) v Amsterdamu[[4]](#footnote-5). Samotné pojmenování pythonu nemá nic společného s druhem hada, neboť Van Rossum pojmenoval python po televizním pořadu anglické BBC Monty Pythonův létající kruh[[5]](#footnote-6).

První verzi kódu zveřejnil Van Rossum v únoru roku 1991 (verze 0.9.0). Již v této fázi vývoje bylo možné v pythonu pracovat s třídy a dědičnostmi. K dispozici již také byli i základní datové typy jako string, list či dict. V počátečním vydání Python již obsahoval modulový systém[[6]](#footnote-7).

Python dosáhl verze 1 v lednu 1994. Novými funkcemi byla lambda[[7]](#footnote-8) a práce s mapováním[[8]](#footnote-9), filtrováním[[9]](#footnote-10) a redukování[[10]](#footnote-11) vyšších funkcí[[11]](#footnote-12) (Funkce vyšších řádů berou jednu nebo více funkcí nebo atributů a jako výsledek vrací jednu funkci). Později Van Rossum opouští CWI a pokračuje na pythonu ve CNRI[[12]](#footnote-13) (Korporace pro národní výzkumné iniciativy) ve Virginii. Další důležitou verzí je 1.4, kde se objevili pojmenované parametry[[13]](#footnote-14) nebo podpora komplexních čísel.

Během Van Rossumova pobytu ve CNRI se institut snažil o zpřístupnění možnosti programování pro veřejnost se základním vzdělání. Tohle mělo později za následek menší "šachování" mezi licencemi pythonu 1.6, které měli za cíl získání licence svobodného softwaru[[14]](#footnote-15) (Free-software licence). Další verze 1.6.1 sice neobsahovala žádné důležité funkce, ale měla změněnou licenci na GPL/GNU[[15]](#footnote-16) (General Public License neboli bezplatná softwarová licence).

Verze 2.0 byla vydána v říjnu 2000, která představila seznamy[[16]](#footnote-17).

2.2 představila sjednocení typů Pythonu (typů napsaných v C) a tříd (typů napsaných v Pythonu) do jedné hierarchie.

2.5 představil prohlášení with[[17]](#footnote-18) umožňující otevření a zavření souboru a další funkce.

Python 2.6 byl vydán, aby se shodoval se souběžným vývojem Pythonu 3.0 a varoval hlavně o funkcích, které jsou ve verzi 3.0 odstraněny.

Podobně byl vydán i 2.7, který se shodoval s 3.1. Python 2.7 byl posledním vydáním ve druhé sérii. V listopadu 2014 byl oznámen konec podpory 2.7 do roku 2020. Uživatelé byli vyzváni, aby postupně přešli na Python 3.0.

1. ledna 2020 byl "zmražen"[[18]](#footnote-19) kód Pythonu 2.7. Konečné vydání, 2.7.18, došlo 20. dubna 2020 a zahrnovalo opravy kritických chyb a blokace vydání.

Verze 3.0 přišla 3. prosince 2008. 3.0 byla navržena, aby napravila základní konstrukční chyby jazyka. Tyhle změny avšak znemožnili zpětnou kompabilitu se staršími verzemi (tudíž došlo k samotnému oddělení verzí z 2.x na 3.0). Hlavním motem změn bylo odstranění nadbytečných, duplicitních konstrukcí a modulů. Vznikl nástroj tzv. 2to3[[19]](#footnote-20), který dokázal přepsat automaticky Python 2 do nové verze, avšak nástroj nefunguje na 100% a některé aspekty nedokáže převést.

Hlavními úpravami byly:

* Změna print, aby se jednalo o vestavěnou funkci. V Pythonu 2.6 a 2.7 print() je k dispozici jako vestavěná funkce, avšak je maskovaná syntaxí příkazu, který lze deaktivovat zadáním „from \_\_future import print\_functionv“ v hlavičce souboru
* Odebrání inputu ve verzi 2. Z verze poté byl přebrat raw\_input, který byl přejmenován na klasický input.
* Přidání podpory pro anotace jednotlivých funkcí (Když v kódu byla nepoužívaná funkce se špatnou syntaxí, tak se program ve 2.x nespustil)
* Sjednocení str / unicode typů
* Odebrání funkcí zpětné kompatibility, včetně tříd starého stylu, výjimek řetězců a implicitních relativních importů
* Změna funkce celočíselného dělení. (ve 2.0 5 / 2 = 2, nyní 5 / 2 = 2.5. ve 3.0 vznikla náhradní syntaxe 5 // 2 = 2)

## wxPython[[20]](#footnote-21)

WxPython je obal pro multiplatformní GUI[[21]](#footnote-22) (grafické uživatelské rozhraní) aplikační programovací rozhraní wxWidgets (napsán v C++[[22]](#footnote-23)) pro programovací jazyk Python. Jedná se o otevřený (veřejný kód) rozšiřující modul Pythonu. Oficiální stránky wxPythonu:

<https://www.wxpython.org>

### Historie wxPythonu

WxPython byl vytvořen Robinem Dunnem[[23]](#footnote-24), když potřeboval GUI k operačnímu systému HP-UX a Windows verze 3.1 (1992-1995). Při hodnocení komečních řešení narazil na vazby Pythonu se sadou nástrojů wxWidgets.

První verze byly vytvořeny ručně. Avšak brzy se kódová základna velmi obtížně udržovala synchronizovaná s novými verzemi wxWidgets. Pozdější verze byly vytvořeny pomocí SWIG [[24]](#footnote-25)(nástroj pro propojení knihoven v C[[25]](#footnote-26) se skriptovacími jazyky), který výrazně snížil množství práce na aktualizaci.

První "moderní" verze 0.3 byla oznámena v roce 1998.

Práce na tutoriálu probíhala ve verzi 4.1.0 s podtitulem „Escaping the Quarantine“.

Celý vývoj wxPython nalezneme na oficiálních stránkách:

<https://wxpython.org/pages/changes/index.html>

## wxWidgets[[26]](#footnote-27)

Samotná sada nástrojů wxWidgets je knihovna nástrojů pro vytváření graficky uživatelských rozhraní napříč všemi platformami. WxWidgets umožňuje kódu GUI kompilovat a spouštět na několika počítačových platformách s žádnými nebo minimálními změnami kódu.

Jedná se o bezplatný a otevřený software[[27]](#footnote-28) distribuovaný za podmínek licence WxWidgets[[28]](#footnote-29), která je obdobná GPL/GNU u Pythonu. Stránky wxWidgets: <https://www.wxwidgets.org/>

### Historie wxWidgets

WxWidgets (původně wxWindows) zahájil v roce 1992 Julian Smart[[29]](#footnote-30) z Edinburské Univerzity[[30]](#footnote-31). V roce 2004 došlo k přejmenování wxWindows v důsledku požadavků společnosti Microsoft pro distribuci v UK.

Hlavní verze byla vydána 6. ledna 2004. Samotná verze 3.0 byla vydána 11. listopadu 2013.

## Jupyter Notebook

Jedná se o webové interaktivní prostředí výpočetní prostředí pro vytváření stejnojmenného dokumentu, který je součástí Projektu Jupyter. Samotný notebook je kombinace Markdown[[31]](#footnote-32) dokumentu formátovaným do JSON[[32]](#footnote-33) (formát slovníku). Zápis do notebooku je prováděn pomocí tzv "buněk", které mohou obsahovat funkční kód mnoha jazyků[[33]](#footnote-34), text, matematiku, grafy, obrázky. Dokumenty mají koncovku "ipynb".

### Historie Jupyter Notebooku

V roce 2014 vznikl Projekt Jupyter oddělením od IPythonu. IPython nadále existuje jako prostředí Pythonu, zatímco Notebook a další jazykově nezávislé části IPythonu se přesunuly pod názvem Jupyter.

V roce 2015 GitHub[[34]](#footnote-35) a projekt Jupyter oznámili nativní vykreslování formátu souborů notebooků Jupyter (soubory .ipynb) na platformě GitHub.

## Elektromagnet ventilProject Jupyter[[35]](#footnote-36)

Obrázek č. 1 - Zobrazení souborů .ipynb na GitHubu

Zdroj: Můj osobní GitHub

Je nezisková organizace vytvořená s motem "rozvinout otevřený software a služby pro interaktnivní práci na počítači přes desitký programovacích jazyků". Jupyter vznikl oddělením od IPythonu v roce 2014, za kterým stál Fernando Pérez[[36]](#footnote-37). Pod Projekt Jupyter spadají interaktivní výpočetní produkty Jupyter Notebook, JupyterHub a JupyterLab.

## IPython[[37]](#footnote-38)

IPython (Interactive Python) je příkazový shell[[38]](#footnote-39) (příkazový řádek pro práci s počítačem) pro interaktnivní výpočty ve více programovacích jazycích (původně pouze pro Python). Podporuje práci s multimédii[[39]](#footnote-40), introspekci[[40]](#footnote-41) (schopnost programu zkoumat typ a vlastnosti objektu při běhu), nástroje pro paralelní výpočty[[41]](#footnote-42) (výpočet komplikovaných příkladů pomocí rozdělení na menší díly, které se vyřeší jednodušeji) a rozhraní poznámkového bloku[[42]](#footnote-43) s podporou kódu, textu, matematických výrazů a dalších médií (podobně jako Markdown).

## NbViewer

Jedná se o nový produkt Jupyter Notebooku, který vytváří rozhraní, které umí vzít veřejné URL[[43]](#footnote-44) notebook dokumentu a převede ho na formát HTML[[44]](#footnote-45), který zobrazí jako webovou stránku.

## GitHub

Obrázek č. 2 - webové rozhraní NbViewer

Zdroj: screenshot z <https://nbviewer.jupyter.org/>

Jedná se o poskytovatele internetového hostingu[[45]](#footnote-46) (webhostingu) určený pro vývoj softwaru[[46]](#footnote-47) pomocí systému pro distribuci Git. Hlavními funkcemi, krom samotného úložiště, je správa zdrojového kódu, systém aktualizací a verzí.

GitHub své služby nabízí zdarma, avšak existují komerční profesionální verze.

Dříve bývali zdarma pouze projekty s otevřeným zdrojovým kódem[[47]](#footnote-48) (veřejným kódem), avšak od roku 2019 GitHub začal postupně rušit omezení pro neplacenou verzi. Nyní může uživatel mít nekonečno tzv. repositářů[[48]](#footnote-49) (úložišť) ať už veřejných, tak soukromých.

Aktuálně GitHub má přes 60 milionů uživatelů[[49]](#footnote-50) a více než 210 milionů úložišť[[50]](#footnote-51), což ho činí největším hostitelem zdrojového kódu na Světě.

### Git[[51]](#footnote-52)

Jedná se o nejpopulárnější systém pro správu verzí pro programátory při vývoji softwaru s GNU licencí.

Nástroj funguje formou kontroly verzí, ve které sleduje změny v celém kódu.

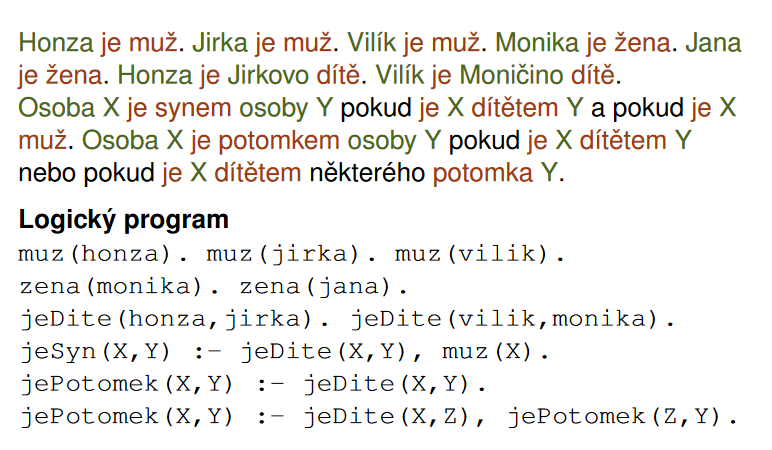
Výhodami samotného Gitu je jeho rychlost a integrita dat[[52]](#footnote-53) a podpora pracovních toků při vývoji (tzv. branche/větve).

## Objektově orientované programování

Téma téhle kapitoly je programovací paradigma OOP.

### Programovací paradigmata[[53]](#footnote-54)

OOP se jedná se o tzv. programovací paradigma. Jedná se o způsob jak rozdělit programovací jazyky. Paradigma se zabývají implikacemi (prováděním) provádění jazyka (samotné psaní kódu). Paradigmata se rozdělují na:

1. Imperativy[[54]](#footnote-55) - uživatel upravuje/instruuje samotné zařízení/stroj, tím mění jeho stav
   * Procedurální programování[[55]](#footnote-56) (Strukturované/imperativní programování): nejzákladnější/nejjednodušší paradigma, které seskupuje pokyny do postupů (jednoduše obsahuje řadu výpočtových kroků, které vedou k cílenému výsledku).
   * Objektově orientované programování (OOP): paradigma je založené na konceptu tzv. "objektů" (viz později).
2. Deklarativní[[56]](#footnote-57) - uživatel pouze upravuje vlastnosti požadovaného výsledku, ale ne způsob jeho výpočtu
   * Funkcionální programování[[57]](#footnote-58): programy jsou konstruovány funkcemi, které se skládají do stromů či samotných výrazů, které vracejí hodnotu podle které se mění stav programu.
   * Logické programování[[58]](#footnote-59): je psán sadami vět v logické podobě, které řeší programové problémy.

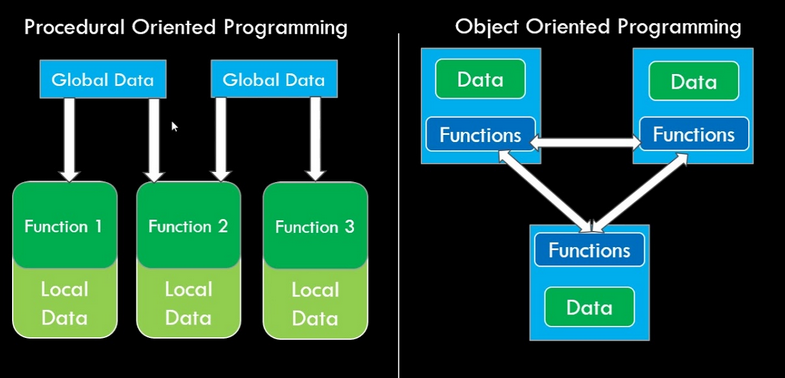
Obrázek č. 3 - Příklad logického programování

Zdroj: prezentace Logické programování - Tomáš Kühr: <https://www.inf.upol.cz/downloads/ruzne/LPproSS.pdf>

* + Matematické programování[[59]](#footnote-60): využívá mnohé způsoby matematických funkcí (lineární, celočíselné, kvadratické, dynamické programování...)

### OOP[[60]](#footnote-61)

OOP paradigma tedy funguje na principů objektů - samotné objekty obsahují buď data[[61]](#footnote-62) (atributy, vlastnosti) nebo funkce[[62]](#footnote-63) (metody).

Funkce objektů spočívá v tom, že objekty mohou přistupovat a upravovat datová pole podle sebe sama (podle jednoho atributu se upraví atribut z jiného objektu). Tudíž objekty mezi sebou interagují.

Jazyků OOP je mnoho. Mezi nejoblíbenější však patří ty, které jsou založeny na třídách[[63]](#footnote-64). Objekty jsou poté vytvářeny do tříd[[64]](#footnote-65), kde se stávají tzv. jejich instancemi[[65]](#footnote-66) (rodiči), které určují jejich datové typy.

Obrázek č. 4 - Rozdíl mezi procedurálním „klasickým“ a objektově orientovaným programováním.

Zdroj: obrázek z videa Simple Snippets o OOP v Javě

Odkaz: <https://www.youtube.com/watch?v=Jt13s8RYAVg>

Mnoho nejrozšířenějších programovacích jazyků (např. C++, Python, Java [[66]](#footnote-67)atd.) jsou založeny na více paradigmech a podporují OOP ve větší či menší míře.

Mezi významné objektově orientované jazyky patří: Java, C++, C#, Python, R, PHP(pro weby), Visual Basic(microsoft), JavaScript(pro weby), Ruby, Pearl, Object Pascal, Objective-C, Dart, Swift, Scala, Kotlin, Common Lisp, MATLAB atd…

### Funkcionalita OOP v Pythonu

V Pythonu se využívá tzv. dědičnost[[67]](#footnote-68) pro opětovné použití kódu a rozšířenost ve formě tříd. V konceptu:

* Třídy - definice datového formátu a dostupné postupy pro daný typ objektu nebo třídy objektu
* Objekt - instance tříd (objekty dědí vlastnosti dané třídy)

Funkce v OOP jsou známe jako metody

Proměnné jako pole[[68]](#footnote-69), členy, atributy nebo vlastnosti.

Existují tedy:

* Proměnné třídy - data patří do třídy jako celku
* Proměnné/atributy instance - data patří pouze jednotlivým objektům
* Členské proměnné[[69]](#footnote-70) - označujeme tak proměnné třídy nebo instance, které jsou definovány konkrétní třídou
* Metody třídy - patří do třídy jako celku a mají přístup pouze k proměnným a volaným vstupům třídy
* Metody instance - patří k jednotlivým objektům, přístup pouze k proměnným konkrétního objektu, na který jsou volány vstupy a proměnné třídy

# Praktická část

Praktická část popisuje výhody využití a aplikaci zmíněných nástrojů v teoretické části a samotnou tvorbu tutoriálu.

## Prostředí Pythonu a Git

Při tvorbě tutoriálu byly využity stabilní verze Pythonu 3.8.6[[70]](#footnote-71) a 3.7.9[[71]](#footnote-72).

Nejnovější verze 3.9.0[[72]](#footnote-73) není zatím kompatibilní s wxPythonem (rok 2020).

Při studování knihovny wx jsem pracoval ve dvou prostředích a to hlavně v Atomu[[73]](#footnote-74) a příležitostně ve Visual Studio Code[[74]](#footnote-75) (VSCode).

Atom je open source IDE navržen týmem GitHubu roku 2014, jehož esenciální součástí jsou stahovatelné balíčky od komunity upravující samotný editor. Atom má v sobě zabudovaný i Git, který jsem, s kombinací GitHub Desktopu[[75]](#footnote-76), aktivně při práci používal.

# Závěr

V této kapitole práce se autor věnuje zhodnocení celé práce. Je vhodné hodnotit práci podle bodů zadání. Uveďte, co jste měli udělat, jak jste to udělali a s jakými výsledky, poznatky, úspěchy či neúspěchy, vyjádřit se k splnění cíle práce. Je nutné zaujmout konkrétní stanovisko k jednotlivým výstupům práce.

Závěr je psán v první osobě jednotného čísla, v minulém čase. Rozsah závěru je jedna až dvě strany.

Nestačí napsat, že se „něco vytvořilo“, ale konkrétně s jakými výsledky, nedokonalosti je potřeba zdůvodnit, uvést možné nápravy, náměty na další práci, výhledy do budoucna atd.

# Seznam použité literatury

Python:

<https://en.wikipedia.org/wiki/Python_(programming_language)>

Poznámky:

[1] definice programovacího jazyka: <https://en.wikipedia.org/wiki/Programming_language>

[2] wiki stránka Guido van Rossuma: <https://en.wikipedia.org/wiki/Guido_van_Rossum>

[3] Bližší informace o CWI: <https://en.wikipedia.org/wiki/Centrum_Wiskunde_%26_Informatica>

[4] televizní seriál Monty Pythonův létající cirkus: <https://en.wikipedia.org/wiki/Monty_Python%27s_Flying_Circus>

<https://cs.wikipedia.org/wiki/Python>

<https://cs.wikipedia.org/wiki/Programovac%C3%AD_jazyk>

https://en.wikipedia.org/wiki/Guido\_van\_Rossum

# Seznam obrázků a tabulek

Zde se vkládá jí seznamy obrázků a tabulek. Vložíte to v menu Odkazy / Vložit seznam obrázků, kde vyberete v Obecných / Popisek titulu Obrázek č. a potom Tabulka č.

[Obrázek č. 1 - Zobrazení souborů .ipynb na GitHubu 12](file:///D:\aŠKOLA\maturitní%20práce\Dokumentace\DPM.docx#_Toc64061122)

[Obrázek č. 2 - webové rozhraní NbViewer 13](file:///D:\aŠKOLA\maturitní%20práce\Dokumentace\DPM.docx#_Toc64061123)

[Obrázek č. 3 - Příklad logického programování 15](file:///D:\aŠKOLA\maturitní%20práce\Dokumentace\DPM.docx#_Toc64061124)

[Obrázek č. 4 - Rozdíl mezi procedurálním „klasickým“ a objektově orientovaným. 16](file:///D:\aŠKOLA\maturitní%20práce\Dokumentace\DPM.docx#_Toc64061125)

[Tabulka č. 1: Měření zesílení operačního zesilovače 3](#_Toc450839160)

# Přílohy

V přílohách se uvádějí složitější obrázky, grafy, schémata zapojení, schémata desek plošných spojů, výpisy programů, fotografie a podobně, které není vhodné dávat do základního textu práce.

Každá příloha je číslovaná a pojmenovaná stejně jako obrázky, včetně zdrojů.

Příloha č. 1: Poster k maturitní práci.

Zde bude **povinně vložen obrázek vytvořeného posteru** ve formátu A4, barevný, orientovaný na výšku popř. i šířku

1. Hlavní zdroj kapitoly - <https://en.wikipedia.org/wiki/Python_(programming_language)> [↑](#footnote-ref-2)
2. Definice programovacího jazyka: <https://en.wikipedia.org/wiki/Programming_language> [↑](#footnote-ref-3)
3. Guido van Rossum:

   <https://en.wikipedia.org/wiki/Guido_van_Rossum> [↑](#footnote-ref-4)
4. Další informace o CWI: <https://en.wikipedia.org/wiki/Centrum_Wiskunde_%26_Informatica> [↑](#footnote-ref-5)
5. Televizní seriál Monty Pythonův létající cirkus -<https://en.wikipedia.org/wiki/Monty_Python%27s_Flying_Circus> [↑](#footnote-ref-6)
6. Metoda funkčnosti programu na nezávislých zaměnitelných modulech - <https://en.wikipedia.org/wiki/Modular_programming> [↑](#footnote-ref-7)
7. Tutoriál na lambdu - <https://www.w3schools.com/python/python_lambda.asp> [↑](#footnote-ref-8)
8. Aplikování funkce na element každé položky v seznamu (functoru) - <https://en.wikipedia.org/wiki/Map_%28higher-order_function%29> [↑](#footnote-ref-9)
9. Funkce zjišťuje datový typ položky a vrací bool hodnotu - <https://en.wikipedia.org/wiki/Filter_%28higher-order_function%29> [↑](#footnote-ref-10)
10. Velmi obsáhlá funkce k uspořádávání a analizování dat - <https://en.wikipedia.org/wiki/Fold_%28higher-order_function%29> [↑](#footnote-ref-11)
11. Definice vyšších funkcí - <https://en.wikipedia.org/wiki/Higher-order_function> [↑](#footnote-ref-12)
12. CRNI - <https://en.wikipedia.org/wiki/Corporation_for_National_Research_Initiatives> [↑](#footnote-ref-13)
13. Pojmenovatelné parametry - <https://en.wikipedia.org/wiki/Named_parameter> [↑](#footnote-ref-14)
14. Licence svobodného softwaru - <https://en.wikipedia.org/wiki/Free-software_license> [↑](#footnote-ref-15)
15. Obecná veřejná licence GNU - <https://en.wikipedia.org/wiki/GNU_General_Public_License> [↑](#footnote-ref-16)
16. Definice seznamu - <https://en.wikipedia.org/wiki/List_comprehension> [↑](#footnote-ref-17)
17. Handler with - <https://en.wikipedia.org/wiki/Python_syntax_and_semantics#With_statements> [↑](#footnote-ref-18)
18. Definice zmražení kódu - <https://en.wikipedia.org/wiki/Freeze_%28software_engineering%29> [↑](#footnote-ref-19)
19. Nástroj 2to3 - <https://docs.python.org/3/library/2to3.html> [↑](#footnote-ref-20)
20. Wiki WxPythonu - <https://en.wikipedia.org/wiki/WxPython> [↑](#footnote-ref-21)
21. Definice GUI - <https://en.wikipedia.org/wiki/Graphical_user_interface> [↑](#footnote-ref-22)
22. Programovací jazyk C++ - <https://en.wikipedia.org/wiki/C%2B%2B> [↑](#footnote-ref-23)
23. Robin Dunn - <https://wiki.wxpython.org/RobinDunn> [↑](#footnote-ref-24)
24. Nástroj SWIG - <https://cs.qaz.wiki/wiki/SWIG> [↑](#footnote-ref-25)
25. Programovací jazyk C - <https://en.wikipedia.org/wiki/C_(programming_language)> [↑](#footnote-ref-26)
26. Wiki WxWidgets - <https://en.wikipedia.org/wiki/WxWidgets> [↑](#footnote-ref-27)
27. Definice softwaru - <https://cs.wikipedia.org/wiki/Software> [↑](#footnote-ref-28)
28. WxWidgets licence - <https://en.wikipedia.org/wiki/WxWidgets#License> [↑](#footnote-ref-29)
29. Julian Smart - <http://www.anthemion.co.uk/julian.htm> [↑](#footnote-ref-30)
30. Edinburská Univerzita - <https://en.wikipedia.org/wiki/University_of_Edinburgh> [↑](#footnote-ref-31)
31. Typ souboru Markdown - <https://en.wikipedia.org/wiki/Markdown> [↑](#footnote-ref-32)
32. Typ souboru JSON - <https://en.wikipedia.org/wiki/JSON> [↑](#footnote-ref-33)
33. Seznam programovacích jazyků - <https://jupyter4edu.github.io/jupyter-edu-book/jupyter.html> [↑](#footnote-ref-34)
34. Wiki GitHubu - <https://en.wikipedia.org/wiki/GitHub> (dále v GitHub kapitole a praktické části) [↑](#footnote-ref-35)
35. Wiki Projektu Jupyter - <https://en.wikipedia.org/wiki/Project_Jupyter> [↑](#footnote-ref-36)
36. Fernando Pérez - <https://en.wikipedia.org/wiki/Fernando_P%C3%A9rez_%28software_developer%29> [↑](#footnote-ref-37)
37. Wiki IPythonu - <https://en.wikipedia.org/wiki/IPython> [↑](#footnote-ref-38)
38. Příkazový řádek - <https://en.wikipedia.org/wiki/Shell_%28computing%29> [↑](#footnote-ref-39)
39. Definice multimédií - <https://en.wikipedia.org/wiki/Interactive_media> [↑](#footnote-ref-40)
40. Prozkoumávání typu nebo vlastností objektu v reálném čase při běhu programu- <https://en.wikipedia.org/wiki/Type_introspection> [↑](#footnote-ref-41)
41. Paralelní výpočty - <https://en.wikipedia.org/wiki/Parallel_computing> [↑](#footnote-ref-42)
42. Rozhraní poznámkového bloku - <https://en.wikipedia.org/wiki/Notebook_interface> [↑](#footnote-ref-43)
43. Jednotná adresa zdroje (Uniform Resource Locator) - <https://en.wikipedia.org/wiki/URL> [↑](#footnote-ref-44)
44. Jazyk pro tvorbu webových stránek - <https://en.wikipedia.org/wiki/HTML> [↑](#footnote-ref-45)
45. Pronájem prostoru na cizím serveru - <https://en.wikipedia.org/wiki/Internet_hosting_service> [↑](#footnote-ref-46)
46. Vývoj softwaru - <https://en.wikipedia.org/wiki/Software_development> [↑](#footnote-ref-47)
47. Veřejně dostupný kód - <https://en.wikipedia.org/wiki/Open_source> [↑](#footnote-ref-48)
48. Repositář - <https://en.wikipedia.org/wiki/Repository_%28version_control%29> [↑](#footnote-ref-49)
49. Počet uživatelů - <https://github.com/search?q=type:user&type=Users> [↑](#footnote-ref-50)
50. Počet úložišť - <https://github.com/search> [↑](#footnote-ref-51)
51. Wiki Gitu - <https://en.wikipedia.org/wiki/Git> [↑](#footnote-ref-52)
52. Integrita (jednotvárnost) dat - <https://en.wikipedia.org/wiki/Data_integrity> [↑](#footnote-ref-53)
53. Hlavní zdroj Programovacích paradigmat - <https://en.wikipedia.org/wiki/Programming_paradigm> [↑](#footnote-ref-54)
54. Definice imperativních paradigmat - <https://en.wikipedia.org/wiki/Imperative_programming> [↑](#footnote-ref-55)
55. Definice procedurálního programování - <https://en.wikipedia.org/wiki/Procedural_programming> [↑](#footnote-ref-56)
56. Definice deklarativních paradigmat - <https://en.wikipedia.org/wiki/Declarative_programming> [↑](#footnote-ref-57)
57. Funkcionální programování - <https://en.wikipedia.org/wiki/Functional_programming> [↑](#footnote-ref-58)
58. Logické programování - <https://en.wikipedia.org/wiki/Logic_programming> [↑](#footnote-ref-59)
59. Matematické programování - <https://en.wikipedia.org/wiki/Mathematical_optimization> [↑](#footnote-ref-60)
60. Wiki OOP - <https://en.wikipedia.org/wiki/Object-oriented_programming> [↑](#footnote-ref-61)
61. Definice dat - <https://en.wikipedia.org/wiki/Data_type> [↑](#footnote-ref-62)
62. Definice funkce/metody - <https://en.wikipedia.org/wiki/Method_%28computer_programming%29> [↑](#footnote-ref-63)
63. Programování založené na třídách - <https://en.wikipedia.org/wiki/Class-based_programming> [↑](#footnote-ref-64)
64. Definice třídy - <https://en.wikipedia.org/wiki/Class_%28computer_programming%29> [↑](#footnote-ref-65)
65. Instance třídy - <https://en.wikipedia.org/wiki/Instance_%28computer_science%29> [↑](#footnote-ref-66)
66. Programovací jazyk Java - <https://en.wikipedia.org/wiki/Java_(programming_language)> [↑](#footnote-ref-67)
67. Dědění funkcí mezi třídami (objekty) - <https://en.wikipedia.org/wiki/Inheritance_%28object-oriented_programming%29> [↑](#footnote-ref-68)
68. Záznamy v tabulce zorganizovány do řádků - <https://en.wikipedia.org/wiki/Field_%28computer_science%29> [↑](#footnote-ref-69)
69. Proměnná spojená čistě s objektem - <https://en.wikipedia.org/wiki/Member_variable> [↑](#footnote-ref-70)
70. https://www.python.org/downloads/release/python-386/ [↑](#footnote-ref-71)
71. https://www.python.org/downloads/release/python-379/ [↑](#footnote-ref-72)
72. https://www.python.org/downloads/release/python-390/ [↑](#footnote-ref-73)
73. https://atom.io/ [↑](#footnote-ref-74)
74. https://code.visualstudio.com/ [↑](#footnote-ref-75)
75. https://desktop.github.com/ [↑](#footnote-ref-76)