Zápisky k předmětu Využití počítačů ve fyzice

Pavel Stránský

15. března 2021

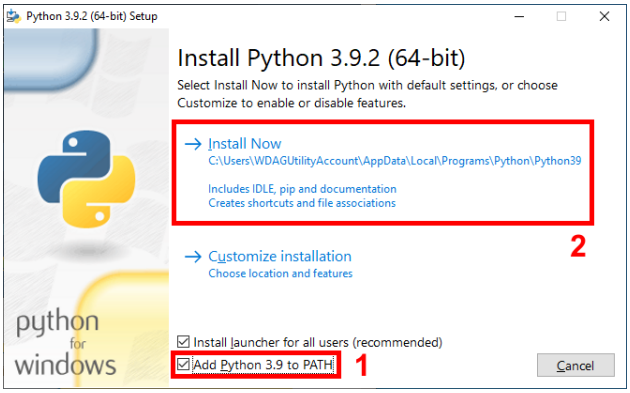
Obsah

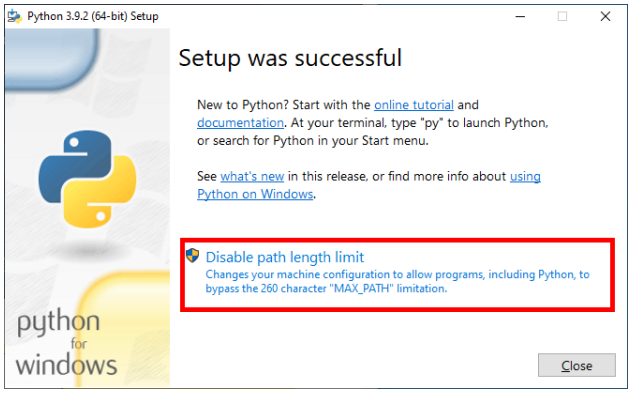
1 Instalace používaných nástrojů

Příklady k cvičení budou demonstrovány v nejnovější verzi programovacího jazyka Python. Jako vývojové prostředí doporučuji Visual Studio Code. Tento volně dostupný program lze nainstalovat na všechny neužívanější operační systémy (Linux, Windows, macOS). Má nepřeberné možnosti při editaci zdrojových souborů, překladu a ladění snad ve všech známých programovacích jazycích. Bohaté možnosti nastavení umožnují přizpůsobit si práci svým potřebám (například zvýrazňování syntaxe, klávesové zkratky či vzhled prostředí). Pomocí doplňků ho můžete integrovat s dalšími službami, například s verzovacím programem Git, či vzdálenými repozitáři, čehož také využijeme. Komunita, která toto vývojové prostředí používá a spravuje, je obrovská, což zaručuje dobrou podporu a rychlé přidávání nových funkcí.

1.1 Instalace Pythonu

Instalační soubor pro svůj operační systém stáhnete ze stránky python.org. Při instalaci na počítač s Windows doporučuji zvolit „Add Python to PATH“, což zjednoduší práci s Pythonem z příkazové řádky, a na poslední obrazovce zvolit „Disable path length limit“:





Pro ověření instalace napíšeme v příkazové řádce příkaz `python`.[1] Tím se spustí REPL[2] Pythonu, ve které můžeme již psát všechny příkazy programovacího jazyka, které se po zadání ihned provedou a vypíšou výsledek.

1.1.1 Instalace doplňujících knihoven

Samotná instalace Pythonu obsahuje jen minimální množství nejnutnějších knihoven. My budeme využívat ještě následující rozšiřující knihovny:

- `NumPy` (**Num**erical **Py**thon: numerická matematika, řady a vícedimenzionální datové typy),
- `SciPy` (**Sci**entific **Py**thon: algoritmy pro optimalizaci, statistiku, řešení diferenciálních rovnic, lineární algebru, atd.),
- `Matplotlib` (vizualizace, grafy).

K jejich doinstalování slouží modul `pip`. V příkazové řádce napíšeme

```
python -m pip install numpy scipy matplotlib
```

čímž se nainstalují naráz všechny tři knihovny.[3]

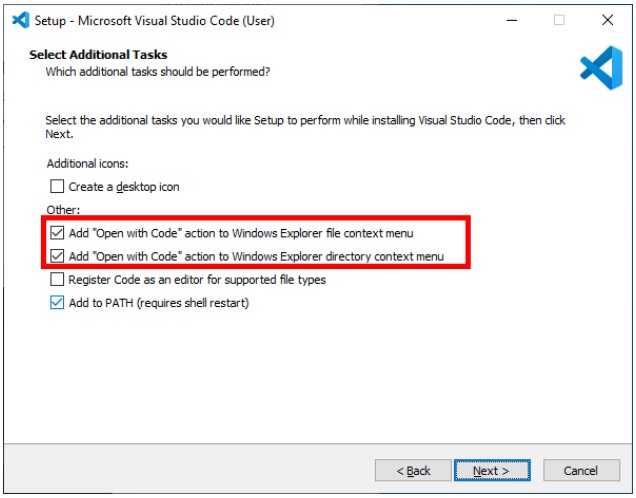
Existuje samozřejmě celá řada dalších užitečných a používaných knihoven, jako je například `Pandas` pro analýzu dat nebo `SymPy` pro symbolické výpočty, na které v tomto kurzu nedojde.

1.2 Instalace Visual Studio Code

Instalace jazyka Python obsahuje jednoduché vývojové prostředí nazvané IDLE.[4] To však poskytuje jen omezené možnosti co se týče ladění, psaní rozsáhlejších projektů s více zdrojovými soubory nebo integrace s verzovacími programy.

Pro serióznější práci budeme používat Visual Studio Code s doplňkem pro programovací jazyk Python. Instalační soubor stáhnete ze stránky code.visualstudio.com. Během instalace na počítač s Windows doporučuji zvolit obě možnosti „Add Open with Code action to...“,

[1] Na počítačích s Linuxem je příkaz v terminálu `python3`.
[2] **R**ead-**E**valuate-**P**rint-**L**oop.
[3] Pokud na počítači s Linuxem uvedený postup nebude fungovat, je potřeba nejprve nainstalovat instalátor `pip` pomocí příkazu `sudo apt install python3-pip`. Pak lze použít buď výše uvedený příkaz, nebo stručnější `pip3 install numpy`.
[4] **I**ntegrated **D**evelopment and **L**earning **E**nvironment.



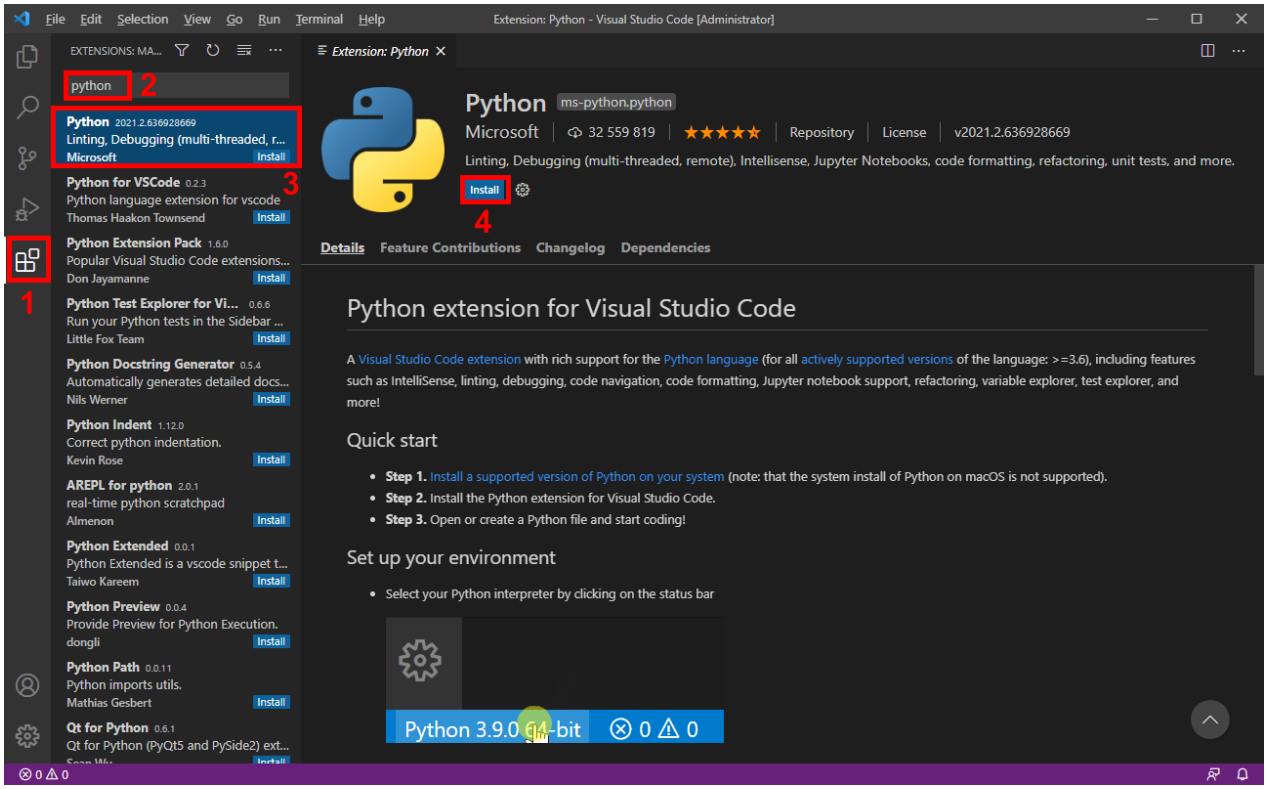
což zjednoduší otevírání složek s projekty nebo samostatných souborů pomocí pravého tlačítka myši.

Na operačním systému Linux je nejjednodušší provést instalci pomocí Snap Store příkazem v terminálu

```
sudo snap install --classic code
```

nebo použít tento návod.

1.2.1 Instalace doplňku pro Python

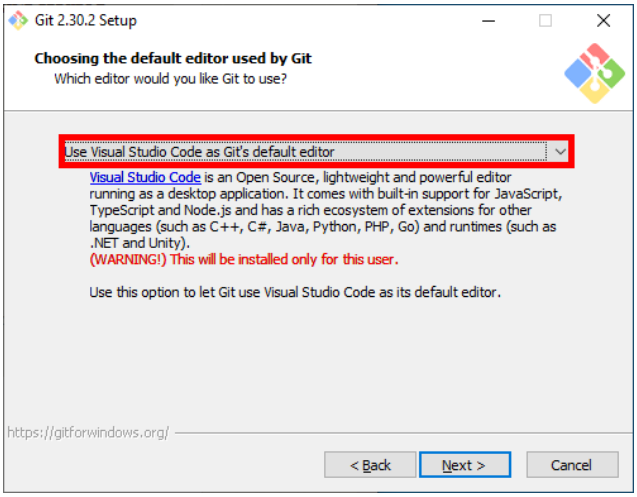


Ke správě doplňků (extensions) se dostanete kliknutím na ikonku 1 nebo stisknutím `Ctrl+Shift+X`. Vyhledáte doplněk Python 2 od Microsoftu, vyberete ho 3 a nainstalujete 4.

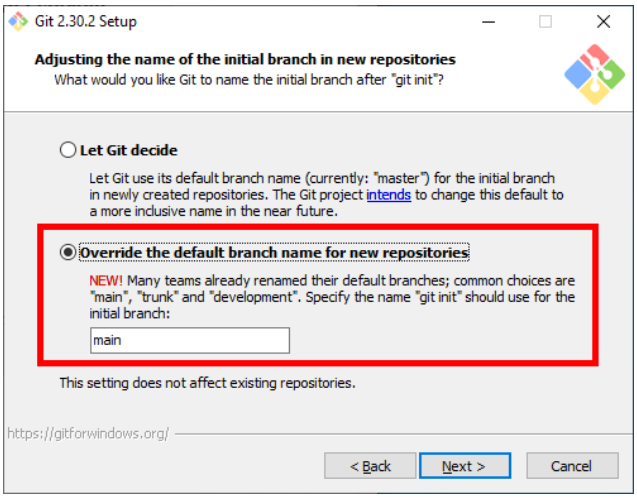
1.3 Instalace Git

Instalační soubor verzovacího systému https://git-scm.com stáhnete z webové stránky git-scm.com. K instalaci Gitu potřebujete administrátorská práva.

V následujícím postupu zobrazuji jen ty snímky obrazovky, na kterých je vhodné vybrat jinou volbu, než jaká je instalátorem standardně nabízena.



Jako editor zvolíme dříve nainstalované Visual Studio Code. Systém Git vyžaduje editor jednak pro povinný komentář každé zapsané změny (commit), jednak pro ošetření kolizí při slučování větví (merge).



Hlavní větev repozitáře se donedávna standardně jmenovala `master`. Vzhledem k negativním konotacím tohoto slova v angličtině se přechází na neutrálnější označení. Nejběžnější je `main`, na které již přešel i populární správce vzdálených repozitářů GitHub. Doporučuji tedy zvolit pojmenování `main`.

Všechna nastavení lze samozřejmě kdykoliv po instalaci změnit.

2 Úvod do systému Git

Git je verzovací systém. Pomáhá udržet historii změn souborů projektu, přičemž ke každému snímku historie se lze vrátit, a pokud se jedná o textový soubor, umí porovnat aktuální a historickou verzi řádek po řádku a zvýraznit provedené změny. Git umožňuje pracovat s nezávislými vývojovými větvemi (*branches*), ve kterých lze například zkoušet různý přístup k řešení daného problému a mezi kterými lze jednoduše přepínat. Nejlepší řešení lze pak snadno začlenit (*merge*) do hlavní vývojové větve.

Git uchovává nejen historické verze souborů, ale také informace o tom, kdo změny provedl. Proto se používá k práci v týmu, kdy každý člen týmu pracuje na určité části projektu a své změny následně do projektu včleňuje. Pod správou Gitu se snadno vyvíjejí i velké projekty, například Visual Studio

Code. Tento projekt je otevřený (open source), což znamená, že kdokoliv, tedy i vy nebo já, má přístup ke všem zdrojovým kódům, může se do práce na projektu zapojit a přispět k jeho vývoji.

Soubory projektu a informace o jejich historických změnách se nazývá souhrnně *repozitář*. Git patří mezi distribuované verzovací systémy. To znamená, že každý uživatel má na svém počítači celý obsah repozitáře. Hlavní výhody tohoto přístupu oproti centrálně řízeným verzovacím systémům jsou dvě:

1. Práce na projektu nevyžaduje připojení k centrálnímu serveru s repozitářem, a tedy můžete pracovat offline klidně někde v džungli nebo na Marsu.

2. Případná porucha počítače spojená se ztrátou dat není žádná katastrofa, protože ostatní členové týmu mají identickou kopii repozitáře na svých počítačích.

Git nejefektivněji funguje na textové soubory, ale zvládne verzovat i soubory binární (například obrázky).

Program Git pochází z prostředí Linuxu, a proto je navržený tak, aby se s ním dalo pracovat z příkazové řádky (terminálu) pomocí jednoduchých textových příkazů. Pro snadnější a rychlejší práci s Gitem je pak dispozici bezpočet různých grafických nadstaveb. Práce s Gitem je dokonce integrována přímo do vývojového prostředí Visual Studio Code.

Na stránkách projektu Git najdete podrobný interaktivní návod ke všem funkcím verzovacího systému (a to částečně i v češtině).

2.1 Prvotní nastavení

Git nelze používat do té doby, než jsou nastaveny základní informace o uživateli. To se provede pomocí příkazů v příkazové řádce (terminálu)

```
git config --global user.name "..."
git config --global user.email "..."
```

přičemž za ... doplníte své jméno (přezdívku) a email. Těmito údaji se podepisují všechny zapsané změny v repozitáři. Pokud tedy spolupracujete na projektu s jinými lidmi, je dobré zvolit takové údaje, pomocí kterých vás kolegové dobře identifikují a snadno kontaktují.

Výpis všech nastavení získáte příkazem

```
git config --global
```

Uvidíte, že v mezi nastaveními je i název hlavní větve repozitáře (`init.defaultbranch=main`) a cesta k nastavenému textovému editoru (v našem případě Visual Studio Code).

Všechna nastavení a práci s příkazem `config` najdete v tomto podrobném návodu nebo zadáním příkazu

```
git help config
```

2.2 Vytvoření lokálního repozitáře

2.3 Cheat Sheet

Pro snadnou orientaci v použití Gitu si můžete vytisknout Git Cheat Sheet nebo využít tento interaktivní web.

3 Úvod do Pythonu

Python je v dnešní době velmi populární programovací jazyk, který pronikl do spousty nesouvisejících odvětví, a to zejména díky bohatosti a pokročilosti knihoven, které jsou vyvíjeny komunitou a jsou tudíž aktuální a dobře odladěné. Python existuje na mnoha platformách od osobních počítačů po mikrokontroléry a dá se v něm naprogramovat téměř cokoliv: pokročilé vědecké výpočty používající nejnovější numerické algoritmy, dobře vypadající grafy, statistická analýza, analýza velkých dat a strojové učení, ale také webové služby, okénkové programy či editace obrázků a videí. Je běžné, že i komerční programy obsahují Python coby skriptovací jazyk a tím umožňují uživateli používat své vlastní kódy „uvnitř“ komerčního produktu.[5]

Python lze charakterizovat jako jazyk:

- *Interpetovaný*, což znamená, že provádění programů probíhá přímo ze zdrojového kódu.[6] Ke spouštění kódů je tedy potřeba mít nainstalovaný interpret, což jsme učinili v sekci 1.1. Pokud kód obsahuje syntaktickou chybu, je odhalena až ve chvíli, kdy se na ni při provádění kódu narazí. U interpretovaných jazyků se neztrácí čas překladem do strojového kódu a je pro ně přirozené dynamické typování proměnných. Tím, že interpet čte text kódu příkaz po příkazu, je provádění programů pomalejší, ale vzhledem k tomu, že velká část časově náročných funkcí a knihoven bývá napsána v rychlejších programovacích jazycích, není to zásadní nevýhoda.

  Opakem intepretovaných jazyků jsou jazyky kompilované, a to buď přímo do strojového kódu daného procesoru[7] nebo do mezikódu, který ke spuštění vyžaduje dodatečnou kompilaci, která však může být vysoce optimalizovaná přímo pro danou hardwarovou konfiguraci použitého počítače.[8]

- *Dynamicky typovaný*, což znamená, že proměnná nemusí mít předem daný typ a typ proměnné se může za běhu programu dokonce měnit.

- *S automatickou správou paměti*, takže programátor se nemusí starat o alokování a uvolňování paměti. Python při inicializaci proměnné paměť automaticky alokuje a ve chvíli, kdy proměnnou přestaneme používat, paměť uvolní.[9]

- V Pythonu lze svým způsobem používat tři základní programovací paradigmata: *procedurální, objektové a funkcionální.*

Python klade důraz na jednoduchost, stručnost a čitelnost kódu. Filosofie programovacího jazyka je shrnuta v Zenu Pythonu.[10]

3.1 Vytvoření a spuštění kódu

Ve Visual Studio Code vytvoříme nový soubor a uložíme si ho s příponou `*.py`. Podle přípony totiž VS Code pozná, že se jedná o pythonovský kód a použije k práci s ním správný doplněk.

Hotový kód spustíte ve VS Code jedním z následujících tří způsobů.

1. *Kliknutím na zelenou šipku na nástrojové liště*: Tímto způsobem spustíte celý soubor kódu.

2. *Stiskem* `F5` (a výběrem Python File): Kód se spustí v režimu ladění (debugger). Zastaví se na každém kontrolním bodě (breakpoint, vloží se na vybranou řádku kódu klávesou `F9`) nebo na každé chybě. Pro pokračování provádění kódu stiskněte buď znovu `F5`, nebo `F10` pro jeden krok.

[5] Jako příklad poslouží nejnovější verze programu Mathematica, Origin či SAS.
[6] Mezi další známé interpetované jazyky patří například JavaScript nebo PHP.
[7] Například C/C++, Pascal nebo Fortran.
[8] Zde se jedná napřílad o jazyky Java, C# a celý .NET framework nebo Julia.
[9] Algoritmus se nazývá *Garbage collection.*
[10] Zen se také vypíše, pokud do svého kódu zadáte `import this`.

3. *Označením části kódu a stiskem Ctrl+Enter*: V okně označeném `TERMINAL` spustí REPL Pythonu a v něm označenou část kódu. REPL se po provedení kódu nezavře, takže v něm lze psát dodatečné příkazy. Pro ukončení REPL do něj stačí napsat `exit()` nebo v něm stisknout `Ctrl+Z` a potvrdit.

3.2 Základy syntaxe Pythonu

Soubor se vzorovými příklady najdete ve vzdáleném repozitáři: `Basics.py`.[11] Doporučuji vám si ho stáhnout nebo jeho obsah překopírovat a důsledně si ho řádek po řádku projít a spouštět nejlépe pomocí označení a stiskem `Ctrl+Enter`, jak bylo popsáno v předchozí sekci. Některé části kódu odkazují na proměnné zavedené v dřívější části kódu, proto kód procházejte postupně od začátku do konce.

Z vzorového souboru bych vypíchl následující body:

1. *Základní datové typy a operátory*: Zaměřte se na možnosti formátování řetězců. Python obsahuje jen dva základní číselné typy: `int` a `float`. Zbytek je podobný jako v jiných programovacích jazycích.

2. *Proměnné a kolekce*: Důležité standardní kolekce jsou *seznam* (list) [...] a *slovník* (dicctionary) {...}. Dále existuje typ *n-tice* (tuple) (...) a *množina* (set) {...}. O jaký typ kolekce se jedná poznáte podle typu závorek, kterými je uzavřena. Python nabízí bohaté možnosti indexování prvků kolekcí.

   Python nemá typ „řada“ (jedno či vícerozměrný soubor hodnot stejných typů). Ten je implementován až v rozšiřujících knihovnách, například v knihovně **`numpy`**, které se budeme věnovat později.

3. *Podmínky a cykly*: Příkaz uvozující blok kódu vždy končí znakem :. Každý blok je definován svým odsazením, které musí být na každém řádku stejné (tj. musí obsahovat stejný počet odsazujících znaků, mezi které patří buď mezera nebo tabulátor). Konvence je používat k odsazení 4 mezery.

   Cykly jsou možné pouze přes iterovatelné objekty. Pro cyklus přes přirozená čísla (indexy) musíme vytvořit odpovídající iterovatelný objekt příkazem `range`. V drtivé většině se lze při programování v Pythonu indexům zcela vyhnout.

4. *Funkce*: Python umožnuje velkou variabilitu co se týče argumentů funkce a návratových hodnot díky své funkci automatického sbalení a rozbalení kolekcí. Funkce se navíc chová jako objekt, lze ji tedy přiřadit jakékoliv proměnné. To je jeden z prvků funkcionálního programování.

   Funkcionální programování také obsahuje koncept anonymní funkce, což je jednoduchá funkce definovaná na jednom řádku, která nemá vlastní jméno. V Pythonu se vytváří pomocí klíčového slova `lambda`.

5. *Moduly*: Při programování je důležité vhodně strukturovat kód. Python obsahuje jednoduchý koncept modulů, přičemž každý soubor s příponou `*.py` lze použít jako modul. Modul je vlastně speciální typ objektu.

   Dobře se seznamte se způsoby, jak modul načíst a používat, jelikož většina funkcí Pythonu se nachází právě v modulech. Jedná se například o matematické funkce v modulu **`math`**.

6. *Třídy*: Python umožňuje objektově orientované programování. Práce s třídami se však v mnohém liší od striktně objektově orientovaných programovacích jazyků, jakými jsou například C++, C# nebo Java. Důležitý rozdíl je například v přístupnosti atributů (všechny atributy v Pythonu jsou veřejné). Rovněž dědičnost a dědění metod se chová odlišně. Dále je nutné pamatovat na to, že každá metoda třídy musí mít v deklaraci jako první argument odkaz na

[11] Vzorový soubor je inspirován příkladem Nauč se Python v Y minutách.

instanci, se kterou je volána (konvenčně se označuje `self`). Ve vzorovém souboru je opravdu jen to nejnutnější minimum. Pokud chcete v Pythonu využívat objekty seriózně, doporučuji pročíst si odpovídající kapitolu v manuálu.

Uvedený soubor s příklady obsahuje jen ty struktury jazyka Python, které budeme používat. V budoucích cvičeních se ještě seznámíte se *správou kontextu* (klíčové slovo `with`). Python umožňuje navíc

- ošetření chyb pomocí *výjimek* (klíčová slova `raise`, `try`, `except`, `finally`),
- tvorbu *generátorů* (klíčové slovo `yield`),
- *asynchronní programování*, korutiny a úkoly (klíčová slova `await`, `async`)
- či tvorbu a použití *dekorátorů*.

Pro plné ovládnutí jazyka vám doporučuji se v budoucnu s těmito koncepty seznámit, a to buď ze specializovaných přednášek, z tutoriálů a návodů na webu, nebo studiem cizích kódů.

3.3 Pojmenování proměnných a formátování

Formátování kódu Pythonu je celkem volné. Povinné je dodržet jen správné odsazení bloků. Pro snazší čitelnost kódu byla nicméně vytvořena řada doporučení, která najdete souhrnně na stránce PEP 8.[12]

Pro pojmenování proměnných existuje jediné závazné pravidlo, které zní, že indentifikátor se nesmí shodovat s žádným z 35 klíčových slov jazyka. Kromě toho jsou ve výše uvedeném dokumentu další nezávazná pravidla k pojmenování proměnných:

- *Proměnné a funkce* se doporučuje pojmenovávat malými písmeny a jednotlivá slova spojovat podtržítky, tzv. underscore style (například `pocet_bodu`).
- *Konstanty* se doporučuje pojmenovávat velkými písmeny a jednotlivá slova spojovat podtržítky (například `PLANCKOVA_KONSTANTA`).
- *Třídy* se doporučuje pojmenovávat tak, že jednotlivá slova názvu začínají velkým písmenem a mezi nimi není žádný znak, tzv. Pascal style (například `PostovniAdresa`).
- Kvůli čitelnosti je dobré se vyhnout jednopísmenným označením `l`, `I` a `O`, jelikož takto označené proměnné mohou být snadno zaměněny s čísly `1` a `0`.

Pokud vám vyhovuje jiný styl pojmenovávání, lze ho použít. Je však dobré být v pojmenovávání konzistentní napříč celým projektem.

Častá otázka je, zda proměnné označovat *česky či anglicky*. Jelikož nikdy nevíte, kdo v dnešním propojeném světě bude chtít váš kód použít a třeba i upravit pro své potřeby, je vhodnější používat anglická pojmenování.

Snadná čitelnost kódu je podpořena i vhodným psaním *komentářů*. Obecné tvrzení zní, že dobře napsaný kód je čitelný i bez komentářů. Poslouží hlavně vhodně označené proměnné a správně navržená struktura kódu. Komentovat je dobré jen ty části kódu, kde se používá nějaký ne všeobecně známý trik nebo postup. Nadbytek komentářů je velmi těžké udržovat při změnách kódu a může vést i k tomu, že po několika úpravách nebudou komentáře v souladu s tím, co kód vykonává.

Naopak dobré je nešetřit vysvětlením, co dělají jednotlivé funkce, jaké jsou jejich argumenty a co vracejí na výstupu. K tomu slouží komentář typu *docstring*, který je vysvětlený v souboru `Basics.py` v sekci o funkcích. Použití *docstringu* usnadní i tvorbu a správu doprovodné dokumentace k celému projektu. Existují například nástroje, které vám z těchto komentářů vytvoří strukturované webové stránky.

[12] PEP = **P**ython **E**nhancement **P**roposal

4 Obyčejné diferenciální rovnice

4.1 Diferenciální rovnice prvního řádu

Nejprve se budeme věnovat řešení jedné diferenciální rovnice prvního řádu,

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(y,t) \tag{1}$$

s počáteční podmínkou

$$y(t_0) = y_0. \tag{2}$$

Zde $y = y(t)$ je hledaná funkce a t je nezávisle proměnná.

Numerické řešení diferenciální rovnice spočívá v nahrazení infinitezimálních přírůstků přírůstky konečnými:

$$\frac{\Delta y}{\Delta t} = \phi(y,t) \tag{3}$$

kde ϕ je funkce, která udává směr, podél kterého se při numerickém řešení vydáme. Volbá této funkce je klíčová a záleží na ní, jak přesné řešení dostaneme a jak rychle ho dostaneme.

4.1.1 Pár důležitých pojmů

- **Explicitní algoritmy:** K výpočtu hodnoty funkce y_{i+1} se vyžadují pouze hodnoty z aktuálních a minulých kroků, tj. y_i, y_{i-1}, atd.

- **Jednokrokové algoritmy:** K výpočtu hodnoty funkce y_{i+1} v následujícím kroku vyžadují pouze znalost hodnoty funkce v aktuálním kroku y_i. Rozepsáním (3) dostaneme

$$\boxed{y_{i+1} = y_i + \underbrace{\phi(y_i,t)}_{\phi_i}\Delta t}, \tag{4}$$

  přičemž počáteční hodnota y_0 je dána počáteční podmínkou. My se omezíme pouze na tyto algoritmy.

- **Lokální diskretizační chyba:**

$$\mathcal{L} = y(t+\Delta t) - y(t) - \phi(y(t),t)\Delta t, \tag{5}$$

  kde $y(t)$ udává přesné řešení v čase t.

- **Akumulovaná diskretizační chyba:**

$$\epsilon_i = y_i - y(t_i) \tag{6}$$

- **Řád metody:** Metoda je p-tého řádu, pokud

$$L(\Delta t) = \mathcal{O}(\Delta t^{p+1}). \tag{7}$$

- **Kontrola chyby řešení:** Chybu numerického řešení diferenciální rovnice lze zmenšit 1) menším krokem, 2) lepší metodou (metodou vyššího řádu). Menší krok však znamená vyšší výpočetní čas. Sofistikované metody proto průběžně mění velikost kroku: když se funkce mění pomalu, krok prodlouží, když se mění rychle, krok zkrátí (tzv. **metody s adaptivním krokem**). Tím se docílí vysoké přesnosti při co nejmenším výpočetním čase.

4.1.2 Eulerova metoda 1. řádu

$$\phi_i = f(y_i, t_i), \tag{8}$$

tj. krok do y_{i+1} děláme vždy ve směru tečny v bodě y_i.

- Nejjednodušší metoda integrace diferenciálních rovnic.
- Chyba je obrovská, k dosažení přesných hodnot je potřeba velmi malého kroku, což znamená dlouhý výpočetní čas.

4.1.3 Eulerova metoda 2. řádu

$$\begin{aligned} k_1 &= f(y_i, t_i) \\ k_2 &= f\left(y_i + k_1 \Delta t, t + \Delta t\right) \\ \phi_i &= \frac{1}{2}\left(k_1 + k_2\right), \end{aligned} \tag{9}$$

tj. uděláme jednoduchý Eulerův krok ve směru k_1, spočítáme derivaci k_2 po tomto kroku a vyrazíme z bodu y_i ve směru, který je průměrem obou směrů (doporučuji si nakreslit obrázek).

Ekvivalentní je udělat „Eulerův půlkrok“ a vyrazit z bodu y_i ve směru derivace spočtené po tomto půlkroku:

$$\begin{aligned} k_1' &= f(y_i, t_i) \\ k_2' &= f\left(y_i + k_1' \frac{\Delta t}{2}, t + \frac{\Delta t}{2}\right) \\ \phi_i &= k_2' \end{aligned} \tag{10}$$

4.2 Runge-Kuttova metoda 4. řádu

$$\begin{aligned} k_1 &= f(y_i, t_i) \\ k_2 &= f\left(y_i + k_1 \frac{\Delta t}{2}, t + \frac{\Delta t}{2}\right) \\ k_3 &= f\left(y_i + k_2 \frac{\Delta t}{2}, t + \frac{\Delta t}{2}\right) \\ k_4 &= f\left(y_i + k_3 \Delta t, t + \Delta t\right) \\ \phi_i &= \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \end{aligned} \tag{11}$$

- Jedna z nejčastěji používaných metod.
- Vysoká rychlost a přesnost při relativní jednoduchosti.
- Existují i Runge-Kuttovy metody vyššího řádu p, avšak vyžadují výpočet více než p dílčích derivací k_j. Obecně platí, že metoda řádu $p \leq 4$ vyžaduje p derivací, metoda řádu $5 \leq p \leq 7$ vyžaduje $p + 1$ derivací a metoda řádu $p = 8, 9$ vyřaduje $p + 2$ derivací.

Úkol 4.1: *Naprogramujte Eulerovu metodu 1. a 2. řádu a Runge-Kuttovu metodu. Vyřešte diferenciální relaxační rovnici*

$$\frac{\mathrm{d}y}{\mathrm{d}t} = -y \tag{12}$$

s počátečními podmínkami $y_0 = 1$ (analytickým řešením je funkce e^{-t}). Integrační krok Δt ponechte jako volný parametr. Nakreslete grafy řešení $y(t)$ pro rozdílné hodnoty integračních kroků, například $\Delta t = 0.01$ a $\Delta t = 0.1$ pro čas $t \in \langle 0; 10 \rangle$.

Úkol 4.2: *Rozšiřte kód tak, aby počítal průměrnou kumulovanou chybu*

$$\mathcal{E} = \sqrt{\frac{1}{n} \sum_{i=0}^{n-1} (y_i - e^{-t_i})^2} \tag{13}$$

a nakreslete závislost $\mathcal{E}(\Delta t)$ pro $\Delta t \in \langle 0.002; 0.1 \rangle$ a pro různé metody. Jelikož očekáváme mocninnou závislost dle (7), kde exponent je tím větší, čím větší je řád metody, je výhodné graf $\mathcal{E}(\Delta t)$ kreslit v log-log měřítku. V Pythonu použijete místo `plot(...)` *funkci* `loglog(...)` *z knihovny* `matplotlib.pyplot`*. Ověřte, že získané křivky jsou v souladu s řády použitých metod.*

Úkol 4.3: *Pomocí naprogramovaných metod vyřešte nelineární diferenciální rovnici*

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \sin(ty) \tag{14}$$

s počáteční podmínkou $y_0 = 1$ a vykreslete graf jejího řešení.