


# Jak programovat v Pythonu pro středně pokročilé

(B3B33LAR – Laboratoře robotiky)

David Koníček  
david.konicek@cvut.cz  
david.konicek@email.cz  
2024 březen/duben

1




## Jak programovat v Pythonu pro středně pokročilé

- Úvod
- Python čistý kód
- Python PEP8
- Komentáře
- Dokumentace v kódu
- Identifikátory
- Struktura souboru
- Autorská práva a licence
- Struktura adresářů
- Nástroje
- Využití AI
- Programovací techniky
- Typové kontroly
- GIT – netradiční pohled
- Testování
- Zdroje, další studium

březen-duben '24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 2

2



## Technické poznámky

- Podklady pokrývají:
  - tematiku požadovanou v rámci předmětu LAR
  - tematiku nepovinnou/nepožadovanou, ale v praxi často používanou
  - materiál obsahuje tedy i mnoho podkladů a podnětů k samostudiu
- Odkazy na externí zdroje jsou provedeny kombinací QR kódu a pod ním schovaného klasický HTML odkazu
- QR kódy odkazy používají důvěryhodné URL zkracovače (<https://tinylinks.cc/>) a odkazy by měly být bezpečné

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 3


3



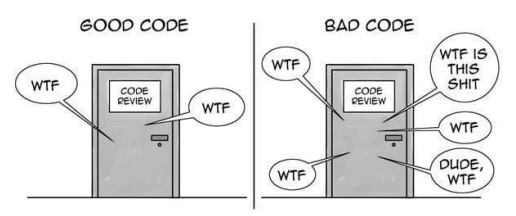
## Motivace k čistému kódu

březen-duben '24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 4

4




## Jak programovat? Čistě!



THE ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 5

5




## Motto (3x)

„Program is written once, but read 50 times“  
*Anonymous*

„Programs are meant to be read by humans and only incidentally for computers to execute.“  
*Donald Knuth*

„80% of the lifetime cost of a piece of software goes to maintenance.“  
*Sun Microsystems: Java Code Conventions (1997)*



březen-duben '24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 6

6

## Jaký kód je čistý?

čistý kód	<i>clean code</i>
čitelný kód	<i>readable code</i>
znovupoužitelný kód	<i>reusable code</i>
udržovatelný kód	<i>maintainable code</i>
v průmyslové kvalitě	<i>industry-strength programming</i>

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 7

7

## Opak čistého kódu

- Nečitelný, špatně udržovatelný, ...
- „**code smell**“
- „**anti-pattern**“

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 8

8

## Výhody a přínosy čistého kódu

- Snadný k porozumění, **co bude kód dělat**
- Snadný k porozumění i **za několik měsíců / roků**
- Snadný k porozumění pro ostatní **uvnitř týmu**
- Snadný k porozumění **mimo tým** (API, knihovny)
- Snadný/jednodušší pro **testování**

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 9

9

## Čistý kód je novinka?

- Vždy byla snaha o správný a čitelný kód
- Rozvoj složitosti programů a řešených oblastí →
- Strukturované programování
- Modulární programování
- Objektové programování
- Čistý kód
- Návrhové vzory a další paradigmaty

Čas ↓ '60, '70, '80, '90-'00, '00-'20

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 10

10

## Programování v 21.století

- **Komplexnost** kódu roste, přesto ...
- **Práce v týmu** (vč. předávání „v čase“ a mezi týmy/projekty), proto ...

... vyvíjený kód musí být:

- **čitelný**
- **pochopitelný**
- **udržovatelný** (včetně hledání chyb i rozvoje kódu)

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 11

11

## Kdy programovat čistě?

- Vždy!**
  - business
  - univerzity
  - open source
- Výjimky:**
  - pokusný a jednorázový kód („trvanlivost“ < 1 den)
  - starý kód
  - výukové materiály pro promítání (prostorová omezení)
  - programátorské soutěže
  - hackaton, workshop, ...

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 12

12

## Jak programovat čistě?

- Změna přístupu:
  - jednotlivec
  - tým
  - management
- Standards a zvyky
- Pravidla a konvence

- Sebedisciplína:
  - jednotlivec
  - tým
  - management
- Pravidla programovacího jazyka
- Nástroje statické kontroly
- Nástroje umělé inteligence (AI)

Co dělat? → Jak to dělat?

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 13

13

## Standards, zvyky, konvence

- Měřítkem "čistoty" je vnímání kódu člověkem ⇒ tedy subjektivní pohled ⇒ **subjektivita** (no)
- Sjednocení pohledu, odstranění subjektivitu ⇒ **standards, zvyky, pravidla a konvence** ⇒ sjednocení pohledu a odstranění duplicity (*good practices*) (yes)

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 14

14

## Univerzálnost pravidel

- Neexistují 100% univerzální pravidla/konvence
- Existují obvykle dodržované pravidla/konvence
- Firmy/projekty/atd. definují vlastní modifikace pravidel
- Dva typické přístupy:
  - Doktrinářský přístup** = žádné výjimky (thumbs down)
  - Volný přístup** = možné výjimky, nutno ale popsat (thumbs up)

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 15

15

## Čistý kód – (některé) oblasti

- Formátování kódu (*style guide*)
- Identifikátory – smysluplné, dobře popisné
- Komentáře a dokumentace
- Uspořádání souborů a adresářů
- Programátorské praktiky – pochopitelný kód

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 16

16

## Čistý kód – (některé) oblasti

- Testy – automatizované
- Pokročilá paradigmatata (koncepty)
- Zpracování chyby, detekce chyb
- GIT, verzování, týmová práce
- Čistá architektura

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 17

17

## Programovací jazyky a čistý kód

Čistý kód je obecný princip  
Čistý kód je aplikovatelný na všechny programovací jazyky

statické	Pascal, C, C++, C#, Java, Ada, TypeScript, Go, ...
dynamické	Python, PHP, JavaScript, R, MATLAB, Lisp, Perl, ...
databázové	SQL, PL/SQL, T-SQL, CQL (Cassandra), GraphQL, ...
skriptovací	Shell (bash, csh, sh), PowerShell, AWK, SED, CMD ...
značkovací	HTML, CSS, XML, ...

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 18


18



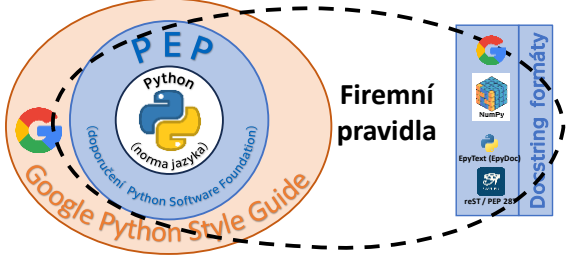
# Python – čistý kód

březen-duben '24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 19

19




## Python čistý kód – oblasti




březen-duben '24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 20

20




## Python – základy čistého kódu

- **PEP 8** - Style Guide for Python Code
- **PEP 484** - Type Hints + **PEP 485** - The Theory of Type Hints
- **PEP 257** - Docstring Conventions
- kvalitně pojmenované **identifikátory**
- čtivé a smysluplné **komentáře**
- **PEP 20** - The Zen of Python
- **PEP 440** - Version Identification




březen-duben '24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 21

21




## PEP = Python Enhancement Proposal

- Původně = návrhy na vylepšení jazyka a knihoven
- Fakticky = schválené vylepšení plus návrhy:
  - **Schválené/aktivní/finální**
  - **Standardizační (standards track)** ... normativní, jsou součástí jazyka a/ nebo standardní knihovny
  - **Procesní (process)** ... řízení procesu schvalování PEP
  - **Informační** historické, atd. ... nejsou normativní
- Dočasně schválené
- Zastaralé
- Otevřené/zvažované
- Odmítnuté/stažené
- **PEP 0** ... seznam všech







březen-duben '24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 22

22




## Další zdroje/inspirace

- **Google Python Style Guide** (github.com, github.io) 
- Reitz & Schlusser: Hitchhiker's Guide to Python Style Guide
- Další (obecná) paradigmatata
- Další programovací jazyky
  - C++
  - Java
  - ...

březen-duben '24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 23

23



## PEP 8 - Style Guide for Python Code

### jako základ pro čistý styl

březen-duben '24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 24

24


## PEP 8 - Style Guide for Python Code

= definice pravidel formátování

- Web [peps.python.org](https://peps.python.org)
- Čitelný web [pep8.org](https://pep8.org)
- Srozumitelné vysvětlení Real Python

• IDE – obvykle silná podpora (dtto výstupy z AI nástrojů)

• Také CLI nástroje (pycodestyle, flake8, ...)



březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 25

25

## PEP 8 základy formátování kódu





březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 26

26

## PEP 8 – znaková sada

- Python pravidla: **UTF-8** (Python 3); ASCII (Python 2)
- Tj. netřeba: `# -*- coding: utf-8 -*-`
- PEP 8 pravidla:
  - pouze anglická abeceda a číslice: **a..z, A..Z, 0..9**
  - speciální znaky: **!"#\$%&'()\*+,-./:;<=>?@[\\^\_`{|}~**
  - mezery (\x20) a konce řádku (\n, \x0A, \r, \x0D)
  - nikdy TAB (\t, \x09) a FormFeed (\f, \x0C, Ctrl + L)
  - nikdy Unicode "exotické" mezery (\x2000, \x200A, atd.)





březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 27

27

## PEP 8 – znaková sada

- Důsledek: „neanglické“ znaky pouze:
  - vlastní jména (lidi, instituce, místa, ...)
  - výstupní text (textové konstanty) pro uživatele
  - textové konstanty pro analýzu dat




březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 28

28

## PEP 8 - odsazování

- Odsazování ... **4 mezery (a násobky)**  
(chytří editor/IDE konvertuje z klávesy TAB)
- Odsazování v rozdělených řádkách – varianty:
  - Zarovnání pod otevírací závorku
  - Odsazení 4/8 mezery
- Nejrychlejší řešení ... automatické formátování:
  - PyCharm : **Ctrl + Alt + L**
  - Visual Studio Code: **Shift + Alt + F** (alt. Ctrl/K, Ctrl/F)

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 29

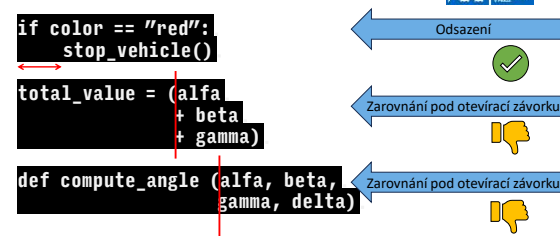
29

## PEP 8 - odsazování

```
if color == "red":
    stop_vehicle()

total_value = (alfa
               + beta
               + gamma)

def compute_angle (alfa, beta,
                  gamma, delta)
```




- Odsazení (green checkmark)
- Zarovnání pod otevírací závorku (thumbs down)
- Zarovnání pod otevírací závorku (thumbs down)

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 30

30


## PEP 8 - odsazování



```

First_angle = compute_angle (
    alfa, beta,
    gamma, delta)


```

← Odsazení (PEP8) 

```

First_angle = compute_angle (
    alfa,
    beta,
    gamma,
    delta)


```

← Odsazení (doporučení) 

```

First_angle = compute_angle (alfa, beta,
                             gamma, delta)


```


← Zarovnání pod otevírací závorku 

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 31

31

## PEP 8 - odsazování



← Dvojnásobné odsazení zalomeného řádku 

```

for count, value in enumerate(
    transform(input_values)):
    print(f"Position {count}: {value}")


```


← Standardní odsazení vnořených příkazů

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 32

32

## PEP 8 – délka řádky






- řádky s kódem <= 79 znaků, Docstring <= 72 znaků
- Alt.: řádky s kódem <= 99 znaků, Docstring <= 72 znaků
- Python Standard Library řádky s kódem <= 79 znaků
- Často výjimky (119 nebo 120 znaků, např. PyCharm) 
- Pozn. pro nastavení: *vertical edge; hard wrap; rules*

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 33

33

## PEP 8 – rozdělování řádky




- Rozdělení řádky ... předchozí řádka končí znakem `\n` 
- Rozdělení řádky ... **otevřené závorky** `(){}[]` (preferované) 
- Pokračovací řádky – varianty pro odsazení:
  - Zarovnání pod otevírací závorku
  - Odsazení 4/8 mezery & otevírací závorka jen s koncem řádky
- Pokračovací řádky ... začínají případným binárním operátorem (*Knuth's method*)

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 34

34


## PEP 8 – rozdělování řádky



```

angles = (alfa,
         beta,
         gamma)


```

← Rozdělení řádky se závorkami 

```

def compute_angle(alfa, beta,
                 gamma, delta):
    ...


```


← Rozdělení řádky zpětným lomítkem 


březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 35


35

## PEP 8 – složené příkazy



- Vždy složené příkazy na více řádků
- Tj. nikdy 2 příkazy na jednu řádku; nikdy `;` 
- Pozn. PEP 8 povoluje výjimku pro jednoduché případy; nedoporučuji

ANO: `if color == "red": stop_vehicle()` 

NE: `if color == "red": stop_vehicle();` 

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 36

36

## PEP 8 – prázdné řádky

- Mezi funkcemi/třídami ... 2 řádky → 3 řádky
- Mezi metodami ... 1 řádka → 2/3 řádky
- Mezi a vnořenými funkcemi ... 1 řádka → 2/3 řádky
- Oddělení úseků souboru ... 1 řádka → 2 řádky
- Oddělení logických úseků kódu ... 1 řádka
- Pro zvýšená čitelnosti kódu ... 1 řádka
- PEP8 povoluje, ale nedoporučuje: Ctrl/L
- **Doporučené výjimky**

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 37

37

## PEP 8 – prázdné řádky

- **Doporučené výjimky**
- Nastavení výjimek v konfigurace IDE nástroje (CLI nástroje):
- PEP 8: **E302** expected 2 blank lines, found 3
- PEP 8: **E303** too many blank lines
- PEP 8: **E305** expected 2 blank lines after class or function definition, found 3

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 38

38

## PEP 8 - Řetězce

- Jednoduché řetězce: uvozovky `"` nebo apostrof `'`
- Doporučuji pro čitelnost: uvozovky `"`
- Víceřádkové řetězce: 3x apostrof `''' text '''`
- Docstring: 3x uvozovky `""" text """`

`""" text """`

**!!!**

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 39

39

## PEP 8 - Mezery ve výrazech a příkazech

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 40

40

## PEP 8 – ANO, používat mezery

- mezeru obecně zvyšuje přehlednost a čtivost
- **za** čárkou `,`, středníkem `,`, dvojtečkou `:`
- **oboustranně** u binárních operátorů

`= + - * / < == > <= >= != : ::`  
`in not in is is not and or not ->`

- složité výrazy – obvykle jen kolem nejméně prioritních operátorů

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 41

41

## PEP 8 – NE, nepoužívat mezery

- na konci řádky
- **za** otevírací závorkou: `( [ {`
- **před** zavírací závorkou: `) ] }`
- před čárkou `y, x = x, y`
- vč. čárky před zavírací závorkou `(1)`
- před středníkem

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 42

42

## PEP 8 – NE, nepoužívat mezery

- před dvojtečkou u příkazů `if x == 4:`
- před dvojtečkou u type hinting `fce(x)`
- před závorkou u volání funkce/metody `array[1:3]`
- před závorkou u indexování a řezů
- vícenásobné mezery

→ výjimka "tabulka" (deklarace/volání funkce)  
 → výjimka "tabulka" (definice strukturovaných konstant)  
 → výjimka "tabulka" type hinting (deklarace funkce)

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 43

43

## PEP 8 – NE, nepoužívat mezery

```
if SERVER == "manual":
    OPCUA_SERVER = "10.100.0.168"
    OPCUA_PORT = "4840"
    OPCUA_USER = "user"
    OPCUA_PASSWORD = "Heslo123+"
    OPCUA_NAMESPACE = "3"
    OPCUA_NAME = ""
    OPCUA_ROOT = "DataMES,"
elif ...
```

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 44

44

## Výjimky z NE, nepoužívat mezery

- ANO, používat mezery: oboustranně u rovnítka `=` pro implicitní hodnoty parametrů funkce s (!!) type hinting
- NE, nepoužívat mezery: oboustranně u rovnítka `=` pro implicitní hodnoty parametrů funkce bez type hinting  
 ⇒ **Doporučená výjimka ... ANO, používat mezery**
- NE, nepoužívat mezery: oboustranně u rovnítka `=` u pojmenovaných argumentů volání funkce  
 ⇒ **Doporučená výjimka ... ANO, používat mezery**

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 45

45

## Výjimky z NE, nepoužívat mezery

```
def opcua_combine_name(
    root_path: str = "\\",
    name_space: str = "ns=2",
    node_short_name: str = "arm",
    delimiter: str = ";"
) -> str:
    ...
```

PEP8

```
def opcua_combine_name(
    root_path: str = "\\",
    name_space: str = "ns=2",
    node_short_name: str = "arm",
    delimiter: str = ";"
) -> str:
    ...
```

doporučená výjimka

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 46

46

## Výjimky z NE, nepoužívat mezery

```
def opcua_combine_name(
    root_path: str = "\\",
    name_space: str = "ns=2",
    node_short_name: str = "arm",
    delimiter: str = ";"
) -> str:
    ...
```

PEP8

```
def opcua_combine_name(
    root_path: str = "\\",
    name_space: str = "ns=2",
    node_short_name: str = "arm",
    delimiter: str = ";"
) -> str:
    ...
```

doporučená výjimka

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 47

47


## PEP 8 – ukončující čárky

- Tj. ukončující čárka před uzavírací závorkou
- Povinně – odlišení závorek u výrazu a inicializaci jednočleného tuple  
`FILES = ("setup.cfg",)`
- Volitelně – definice seznamů, tuple, slovníků, parametrů, ale musí být víceřádkové

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 48

48






# Komentáře (nejen podle PEP 8)

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 49

49




## Kdy použít komentáře

- Dokumentace kódu = docstring (soubory, třídy, metody/funkce)
- Zvýšení čitelnosti kódu
- Nejasný kód; obtížně pochopitelný kód
- Speciální příznaky (TODO, atd.)
- Pokyny/příkazy pro IDE, kontrolní nástroje, atd.

- **Ideálně:** Docstring + speciální příznaky + pokyny pro IDE
- **Ideálně:** ostatní komentáře jen střídmě, platí „méně je více“

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 50

50




## Kdy nepoužít komentáře

- Nikdy – “odstranění” kódu, “zakomentování” kódu; výjimky: pokusný, vyvíjený nebo a jednorázový kód („trvanlivost“ < 1 den)
- Neudržované komentáře nerespektující úpravy kódu (!!!)
- Nekomentujte to, co je přímo v příkazu.

NE: `# The total number of elements is count + 1.`  
`count = count + 1`

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 51

51




## Jak redukovat komentáře

- **Refaktoring identifikátorů / názvů:**  
Často lze nahradit komentář vhodným názvem proměnné / funkce / metody.
- **Refaktoring struktury kódu:**  
Často lze nahradit komentář voláním vhodně pojmenované (nově vytvořené) funkce / metody realizující celý blok komentovaného kódu.

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 52

52




## Jak psát komentáře

- **Anglicky**
- Gramaticky správně včetně spellingu, gramatiky a interpunkce.
- ⇒ Komentář musí být normálně čitelný anglický text.

- AI nástroje:
  - podporují tvorbu komentářů + Docstring
  - nutno kontrolovat
  - často nutno „dočistit“

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 53

53

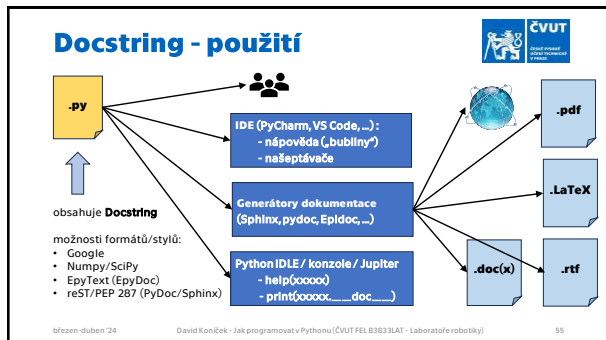


## Python komentáře

- **Docstring** = technická dokumentace uvnitř kódu
- **Komentáře v kódu** (zvýšení čitelnosti kódu, ...)
  - blokové
  - jednořádkové
  - popř. obsahují speciální příznaky (TODO, atd.)
- **Speciální komentáře**
  - příkazy (obvykle jednořádkové; přesná syntaxe)

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 54

54



55

## Docstring komentáře

- Docstring = **technická dokumentace uvnitř kódu**:
  - balíčky (packages)
  - moduly / soubory
  - programy / skripty
  - třídy
  - metody / funkce
- Docstring:
  - **jednořádkový**: pro pomocné třídy / metody / funkce
  - **víceřádkový**: pro veřejné (popř. hlavní) části kódu

březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 56

56

## Docstring komentáře

- PEP 257 – Docstring konvence
  - určuje **obsah**, neurčuje přesný formát
- Docstring podléhají pravidlům odsazování.
- PEP8 Docstring: 3x uvozovky `""" text """` (!!!)

```
"""
text
"""
```

březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 57

57

## Docstring formát

- Jednořádkový: `"""short-description"""`
- Víceřádkový (obvyklejší):

```
"""
one-line summary line
... other information
"""
```

← Prázdná řádka za první řádkou

← Prázdná řádka na konci

březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 58

58

## Metody a funkce - Docstring

- **stručný popis**
- **úplný popis**
- **parametry**
  - jméno                    • typ                    • popis
  - jsou-li nepovinné [optional]    • implicitní hodnoty
  - možnost použití klíčového slova [keyword]
- **návratová hodnota** (typ, popis)
- **výjimky** (vznikající v metodě/funkci)

březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 59

59

## Třída - Docstring podle PEP 257

- **stručný popis**
- **úplný popis** (u podtříd uvést nadtřidu a popsat rozdíly)
- **veřejné atributy** (jméno, typ, popis)
- **veřejné metody třídy** (u podtříd uvádět pro zděděné metody, popsat rozdíly a uvést jsou-li override nebo extend)
- Pozn. konstruktor třídy se dokumentuje u metody `__init__()`

březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 60

60

## Modul/soubor – Docstring (PEP 257)

- **stručný popis**
- **úplný popis**
- **veřejné třídy** (jednořádkový popis, detailně až u implementace)
- **veřejné výjimky** (jedna řádka popis, detailně až u implementace)
- **veřejné funkce** (jedna řádka popis, detailně až u implementace)
- **veřejné konstanty a globální proměnné (jméno, typ, popis)**

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 61

61

## Balíček [package] – Docstring (PEP 257)

- **stručný popis**
- **úplný popis**
- **veřejné vnořené moduly**
- **veřejné vnořené balíčky** [subpackage]
- **veřejné třídy** (jednořádkový popis, detailně až u implementace)
- **veřejné výjimky** (jednořádkový popis, detailně až u implementace)
- **veřejné funkce** (jednořádkový popis, detailně až u implementace)
- **veřejné konstanty a globální proměnné (jméno, typ, popis)**
- uvádí se v modulu `__init__.py` celého balíčku

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 62

62

## Program/skript – Docstring (PEP 257)

- Cca shodný s textem, který program vytiskne jako svůj návod k použití [help]; může být dlouhý i několik obrazovek
- Podrobný natolik, aby i nový uživatel použil program správně
- Kompletní popis všech parametrů [quick reference]
- **stručný popis**
- **úplný popis**
- **syntaxe a parametry příkazové řádky**
- **používané proměnné prostředí operačního systému**
- **používané konfigurační soubory**

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 63

63





## Docstring - další (nepovinné) údaje

- vedlejší efekty (side-effect)
- omezení při použití
- popis implementace
- použité zdroje, informace, atd.

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 64

64

## Docstring komentáře - formáty

- Google 
- NumPy / SciPy 
- EpyText (EpyDoc) 
- reStructuredText (reST/PEP 287/Sphinx) 

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 65

65

## DocString formáty

- **Google**: několik sekcí  
jméno sekce : \n – odsazení – jméno (typ): – popis
- **NumPy / SciPy**: několik sekcí:  
jméno sekce – \n podtržení pomlčkami \n  
jméno parametru, atd. – typ – odsazení – popis
- **reStructuredText (reST/PEP 287/Sphinx)**: několik řádků  
: klíčové slovo – mezera – jméno : popis nebo typ
- **Epytext (EpyDoc)**: několik řádků, podobně JavaDoc  
@klíčové slovo/pole – mezera – jméno : popis nebo typ



**Bez možnosti formátování**

**Markup jazyky = možnost bohatého formátování**

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 66

66

## Google Docstring příklad

```
def calculate_area(radius):
    """
    Calculates the area of a circle.

    Args:
        radius (float): The radius of the circle.

    Returns:
        float: The area of the circle.



    Raises:
        ValueError: If the radius is negative.

    Examples:
        >>> calculate_area(5)
        78.53981633974483
        """
    if radius < 0:
        raise ValueError("Radius cannot be negative.")
    return 3.14159 * radius ** 2
```

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 67

67

## Google sekce

- Stručný a detailní popis modulu / třídy / metody / funkce


**Args:** jméno argumentu, typ, popis  
**Returns:** typ a popis návratové hodnoty  
**Yields:** typ a popis návratové hodnoty  
**Raises:** seznam výjimek a důvod vzniku

**Classes:** jména a krátký popis třídy  
**Methods:** jména a krátký popis metod třídy  
**Functions:** jména a krátký popis metod funkce  
**Attributes:** jméno atributu, typ, popis

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 68

68

## Google sekce



**Examples:** (nepovinné) ukázky použití (volání)

**See Also:** (nepovinné) související třídy, typy, funkce, atd.



**Note:** (nepovinné) další poznámky

**References:** (nepovinné) odkazy na externí zdroje

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 69

69

## Numpy Docstring příklad

```
def calculate_area(radius):
    """
    Calculate the area of a circle.

    Parameters
    -----
    radius : float
        The radius of the circle.

    Returns
    -----
    float
        The area of the circle.



    Raises
    -----
    ValueError
        If the radius is negative.

    Examples
    -----
    >>> calculate_area(5)
    78.53981633974483
    """
    if radius < 0:
        raise ValueError("Radius cannot be negative.")
    return np.pi * radius ** 2
```

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 70

70

## NumPy/SciPy sekce





**Parameters** jméno argumentu, typ, popis  
**Returns** typ a popis návratové hodnoty  
**Yields** typ a popis návratové hodnoty  
**Raises** seznam výjimek a důvod vzniku  
**Attributes** jméno atributu, typ, popis  
**Methods** jména a krátký popis metod třídy

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 71

71

## NumPy/SciPy sekce



**See Also** (nepovinné) související třídy, typy, funkce ...  
**Notes** (nepovinné) další poznámky  
**Examples** (nepovinné) ukázky použití (volání)  
**Warnings** (nepovinné) upozornění spojená s použitím

- Řada dalších direktiv (struktura textu, formátování textu)
- Podpora LaTeX-u

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 72

72

## reST Docstring příklad

```
def calculate_area(radius):
    """
    Calculate the area of a circle.



    :param radius: The radius of the circle.
    :type radius: float

    :return: The area of the circle.
    :rtype: float

    :raises ValueError: If the radius is negative.

    :Example:

    >>> calculate_area(5)
    78.53981633974483
    """
    if radius < 0:
        raise ValueError("Radius cannot be negative.")
    return 3.14159 * radius ** 2
```





březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 73

73

## reST/PEP 287/Sphinx klíčová slova

- :param** jméno a popis argumentu
- :type** typ parametru
- :return** popis návratové hodnoty
- :rtype** typ návratové hodnoty
- :raises** seznam výjimek a důvod vzniku
- :func** popis funkce
- :method** popis metody
- :ivar** jméno a popis proměnné / atributu
- :ivartype** typ proměnné / atributu



březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 74


74

## reST/PEP 287/Sphinx klíčová slova

- :example** (nepovinné) ukázky použití (volání)
- :note** (nepovinné) další poznámky
- :warning** (nepovinné) upozornění spojená s použitím
- :todo** (nepovinné) seznam dalších prací
- :seealso** (nepovinné) související třídy, typy, funkce ...

\* ... a další

- Bohaté možnosti formátování, podobné Markdown
- Řada dalších direktiv (struktura textu, formátování textu)



březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 75

75

## EpyText Docstring příklad

```
def calculate_area(radius):
    """
    Calculate the area of a circle.


    @param radius: The radius of the circle.
    @type radius: float

    @return: The area of the circle.
    @rtype: float

    @raise ValueError: If the radius is negative.

    @example:

    >>> calculate_area(5)
    78.53981633974483
    """
    if radius < 0:
        raise ValueError("Radius cannot be negative.")
    return 3.14159 * radius ** 2
```



březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 76

76

## Epytext klíčová slova [field / tag]

@param	@requires	@summary	@author
@type	@precondition	@todo	@contact
@return	@postcondition	@note	@organisation
@rtype	@invariant	@attention	@copyright
@keyword	@deprecated	@bug	@license
@raise	@permission	@see	@version
	@since	@status	

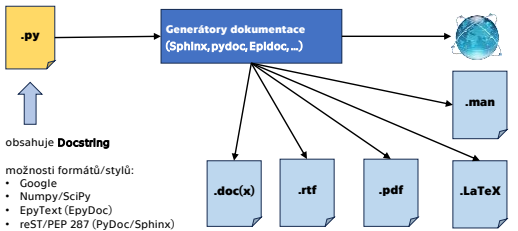



• Bohaté možnosti formátování, podobné, ale slabší, než Markdown nebo reST

březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 77

77

## Docstring - použití



obsahuje **Docstring**

možnosti formátů/stylů:

- Google
- NumPy/SciPy
- EpyText (EpyDoc)
- reST/PEP 287 (PyDoc/Sphinx)

březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 78

78

## Generátory dokumentace

- Sphinx <https://www.sphinx-doc.org/>
- Epydoc <https://epydoc.sourceforge.net/>  
už není udržován (od 2013)
- Doxygen <https://www.doxygen.nl/index.html>
- + doxypypy <https://github.com/Feneric/doxypypy>
- MkDocs <https://www.mkdocs.org/>
- <https://pypi.org/project/mkdocs/>
- pdoc <https://github.com/mitproxy/pdoc>
- pdoc3 <https://pdoc3.github.io/pdoc/>
- pydoctor <https://github.com/twisted/pydoctor>

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 79

79

## Dokumentace projektů/balíčků

- dokumentace vně kódu
- Adresářová struktura více souborů; existuje více variant
- Obsahuje:
  - dokumentaci
  - instalační návody
  - popř. testy, licence, návody, příklady použití
- Realizace:
  - adresář
  - GIT repozitář
  - distribuční balíček [package]

**Viz dále - struktury adresářů, projektů a balíčků**

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 80

80

## Python komentáře

- **Docstring** = technická dokumentace uvnitř kódu
- **Komentáře v kódu** (zvýšení čitelnosti kódu, ...)
  - blokové
  - jednořádkové
  - popř. obsahují speciální příznaky (TODO, atd.)
- **Speciální komentáře**
  - příkazy (obvykle jednořádkové; přesná syntaxe)

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 81

81

## PEP 8 - komentáře

- **Blokové**
  - # + **mezera** + komentář
  - na samostatném řádku (!!!)
  - předchozí řádek před příslušným příkazem (!!!)
  - obvykle jeden prázdný řádek před komentářem
- **Jednořádkové** (inline)
  - nejméně **2x mezera** + # + **mezera** + komentář
  - na konci řádku

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 82

82

## Blokové a inline komentáře

```
# We use a weighted dictionary search to find
# out where i is in the array. We extrapolate
# position based on the largest number
# in the array and the array size and then
# do binary search to get the exact number.
```

```
if i & (i-1) == 0: # True if it's 0 or power of 2.
```

**2x mezera**

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 83

83


## Python komentáře

- **Docstring** = technická dokumentace uvnitř kódu
- **Komentáře v kódu** (zvýšení čitelnosti kódu, ...)
  - blokové
  - jednořádkové
  - popř. obsahují speciální příznaky (TODO, atd.)
- **Speciální komentáře**
  - příkazy (obvykle jednořádkové; přesná syntaxe)

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 84

84

## Speciální komentáře



A. **speciální komentáře** [někdy též codetags]


- Pozn. existuje odmítnuté doporučení PEP 350

B. **řídící příkazy** pro systémy statické analýzy, IDE, atd.

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 85

85

## „Řídící“ komentáře pro IDE, atd.




**#NOSONAR**  
**#NOINSPECTION** xxx ... PyCharm  
<https://www.jetbrains.com/help/pycharm/disabling-and-enabling-inspections.html#comments-ref>  
**# pylint: disable=xxx**  
[https://pylint.pycqa.org/en/latest/user\\_guide/messages/messages\\_\\_overview.html](https://pylint.pycqa.org/en/latest/user_guide/messages/messages__overview.html)  
**# pyright: ignore [xxx]** Pylance/Pyright  
<https://github.com/microsoft/pyright/blob/main/docs/comments.md>  
<https://github.com/microsoft/pyright/blob/main/docs/configuration.md>  
**# type: ignore [xxx]** ... MyPy / PEP484  
[https://mypy.readthedocs.io/en/stable/error\\_code\\_list.html](https://mypy.readthedocs.io/en/stable/error_code_list.html)  
[https://mypy.readthedocs.io/en/stable/error\\_code\\_list2.html#error-codes-optional](https://mypy.readthedocs.io/en/stable/error_code_list2.html#error-codes-optional)

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 86

86

## Speciální příznaky [codetags]




<b># TODO</b>	... popis toho, co je nutné dodělat
<b># FIXME</b>	... popis <b>interní</b> chyby a co je nutné opravit
<b># REALLY</b>	... zdůraznění, že <b>neobvyklý</b> kód je určitě správně
<b># MAGIC</b>	... triky, obtížné na pochopení, atd.
<b># HACK</b>	... popis speciálních postupů
<b># WORKAROUND</b>	... popis <b>externí</b> chyby a jejího napravení
<b># REQUIRES</b>	... popis, co je vyžadováno nad rámec kódu
<b># ASSUMPTION</b>	... popis, co kód předpokládá, obvykle o proměnných
<b># SIDE-EFFECT</b>	... používaný nebo využívaný vedlejší efekt
<b># DIRTY</b>	... opravdu nevhodný kód, který je ale nezbytný

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 87

87

## Komentář #TODO




- popis toho, co je nutné dodělat (doprogramovat)

```
color = WHITE # TODO: add more functionality
```

alternativně:

```
color = WHITE # TODO: add color selection
```


- Často podpora v IDE
- možné kombinovat s odkazem na systém sledování chyb (Atlassian Jira, BugZilla, ...), tedy ID příslušného záznamu
- možné kombinovat s iniciálami autora, datem, atd.



březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 88

88

## Komentář #TODO



- možné kombinovat s výjimkou `NotImplementedError`

```
def set(self, channel): # TODO: add implementation
    raise NotImplementedError("Can't use 'set'.")
```

- Správné použití této výjimky je jen v abstraktní metodě (hlavní) třídy, lze ale použít i takto explicitně.
- Vždy uvádět smysluplný parametr (tj. text)!

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 89

89

## Komentáře #FIXME a #REALLY



**# FIXME:** blabla ... popis **interní** chyby a popř. toho, co a jak je nutné opravit; někdo používá alternativně jako popis nutného zlepšení (nikoliv chyby)

**# REALLY:** blabla .... krátce, že kód je vlastně správně (ačkoliv kód vypadá, že je chybný, resp. podezřelý)

```
except Exception: # REALLY: catch all exceptions

pass # REALLY: do nothing
```

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 90

90

## Komentáře – speciální příznaky

**# MAGIC:** blabla .... triky, obtížné na pochopení, atd.

**# HACK** (for ... browser/OS/library ...): blabla... popis speciálních postupů .... (nebo nic, je-li to jasné z kódu)

**# WORKAROUND** (for ... what ...): ... popis chyby a jejího napravení ... (nebo nic, je-li to jasné z kódu)

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 91

91

## Komentáře – speciální příznaky

**# REQUIRES:** blabla ... popis, co je vyžadováno (další moduly, nastavení OS, data, další programy, atd.)

```
# REQUIRES installation of module poppler
import pdfplumber
```

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 92

92

## Komentáře **#ASSUMPTION**

**# ASSUMPTION:** blabla ... popis, co kód předpokládá, obvykle o hodnotách proměnných, parametrů, mezivýsledků, atd.

```
#ASSUMPTION: velocity is positive
```

- alternativně – zahrnout do komentáře celé funkce / metody / třídy / modulu).

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 93

93

## Komentáře **#ASSUMPTION**

- lépe – použít příkaz **assert**

```
assert velocity > 0, f„bad value of velocity (={velocity})“
```

- **Předpoklad** [příkaz **assert** nebo **#ASSUMPTION**]
  - vyžadovaný logikou výpočtu
  - by měl být vždy splněn, tedy vyloučení nemožných situací.
- **Výjimka** [exception]
  - chybový stav, který může někdy nastat.
  - tj. příkazy a objekt: **raise / try / Exception** ]

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 94

94

## Komentáře **#SIDE-EFFECT** a **#DIRTY**

- Nejlepší je psát kód bez vedlejších efektů a jiného nevhodného řešení. Nelze-li jinak, nutno precizně okomentovat.

**# SIDE-EFFECT:** blabla .... popsat používaný nebo využívaný vedlejší efekt deklarované nebo používané funkce

**# DIRTY:** blabla .... Zdůraznit a komentovat opravdu nevhodný kód, který je ale nezbytný

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 95

95

## Jmenné konvence, identifikátory, atd. (nejen PEP 8)

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 96

96



## PEP 8 – jmenné konvence

- Popisné (!!!) identifikátory
- Povinně ASCII, resp. **písmena, číslice, pomlčka**
- **Celá slova**; jen výjimečně zkratky a akronymy
- Preferovaně **anglicky**
  - včetně spellingu
  - obvykle bez gramatické preciznosti (členy, "to", "of", "s")
- **Popis podle funkcionality**

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 97

97

## Nevhodné pojmenování

- Nepopisovat podle implementace
- Nepopisovat podle návratového typu (tzv. maďarská notace)
- Nikdy nepoužívat matoucí/zaměnitelná slova znaky:
  - klíčová slova
  - velké písmeno I (tedy „íííí“)
  - malé písmeno l (tedy „el“)
  - velké písmeno O (tedy „óóó“)

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 98

98



## Nevhodná jména

- Jednopísmenné názvy, zkratky, akronymy
- Nevyslovitelná a nesmyslná slova včetně zkratk
- Neslušná slova (pozn. kulturně závislé)
- Slangová slova
- Názvy obvyklých knihoven a jejich častých funkcí
- Matoucí název neodpovídá významu

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 99

99

## PEP 8 - Jmenné konvence - styly

Styl jména	Entity jazyka Python	
lowercase lowercase_name 	proměnné funkce moduly / soubory	atributy (tříd) metody (tříd)
CAPITAL CAPITAL_LETTERS	konstanty	
Capitalised CapitalisedWords 	třídy	typy
CapitalisedWordsError	výjimky	
lowercasename	balíčky / adresáře	

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 100

100

## PEP 8 – jmenné konvence

- **Speciální globální proměnné** („dunder“) `__xyz__`
- **Speciální metody tříd** („dunder“) `__xyz__`
- **Privátní metody a členy tříd a modulů** `_alfa`
- **Metody instancí – první argument** `self`
- **Metody tříd – první argument** `cls`
- Výjimky z jmenných konvencí – starý kód nebo kompatibilita s implementací v jiném jazyce (PIL, logger, ...)

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 101

101

## Proměnné, atributy, třídy

- **Podstatné jméno**, případná přičestí následují; bez členů
- **Číselné hodnoty** ⇒ nejlépe počítatelná podstatná jména nebo jasná „číselná“ sémantika:
  - `age` `year` `item_count` `order_number` `arm_length` `wheel_diameter` ...
- **Řetězcové hodnoty** nejlépe:
  - `xxx_name` `xxx_title` `message_xxx_name` ...

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 102

102

## Funkce a metody - konvence

- **Sloveso**, nejlépe akční sloveso:  
`load_xxx` `fetch_xxx` `run_xxx` `compute_xxx` `move_xxx` ...
- **Predikáty**, nejlépe sloveso navozující odpověď ANO/NE:  
`is_xxx` `has_xxx` `was_xxx` `can_xxx` ...
- **Atributy metod**, slovesa get, set, has:  
`get_xxx` `set_xxx` `has_xxx` ...
- Konzistentní použití všech slov (!!!)

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 103

103

## Matematika, fyzika, ...

- Obvykle pracují s jednopísmennými proměnnými
- Dále si "pomáhají" dalšími abecedami, indexy, ...
- Nahradit plnohodnými identifikátory
- Fyzikální jednotky ... vše v jednotkách podle SI, jinak jednotku uvést v názvu, obvykle na konci identifikátoru

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 104

104

## Jmenné konvence - výjimky

- Zkratky a akronymy, které jsou opravdu široce známé, popř. normalizované (ISO/EN/ČSN):
  - měny, státy, ekonomika: `czk` `eur` `vat` `kpi` `id` ...
  - IT (`tcp` `udp` `url` `http` ...)
  - aplikační doména
- Jednopísmenné/zkrácené v opravdu krátkém (!!!) kódu :
  - pro indexaci bez dalšího významu (`i` `j` `k` `x` `y` ...)
  - pomocné proměnné (`tmp` `txt` `len` ...)

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 105

105

## Příkaz import, použití modulů a balíčků

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 106

106

## Moduly a balíčky

- **Modul** v Pythonu:
  - Rozdělení problému na menší části
  - Znovupoužitelný kód
  - Využití kódu jiných autorů
- **Balíček** (*package*) v Pythonu:
  - více modulů (!!!), popř. více „pod-balíčků“
  - tečková notace
  - adresářový strom
  - povinně soubor `__init__.py`
  - použití: distribuce kódu, knihovny, API, ...

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 107

107

## Moduly a balíčky

- **Technické varianty modulu** v Pythonu:
  1. Python soubor
  2. C modul
  3. Vestavěné (built-in) moduly
- Příkaz `import` – zpřístupnění obsahu modulů + balíčků:
  - Moduly – hledá soubory
  - Balíčky – hledá adresáře se souborem `__init__.py`  
– pak hledá soubory a podadresáře

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 108

108

## Příkazy `import` – pořadí hledání

- moduly již v paměti (proměnná `sys.modules` )
- Aktuální adresář
- PYTHONPATH proměnná operačního systému
- Pevná místa instalace Python (podle operačního systému)
- Pozn: Anaconda a VENV (Virtual Environments) mění tyto cesty

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 109

109

## Pevná místa instalace Python

- Windows
  - `c:\Program Files\PythonXY\Lib\`
  - `c:\Users\USERNAME\AppData\Roaming\Python\PythonXY\site-packages`
- Linux
  - `/usr/local/lib/pythonX.Y/site-packages`
  - `/usr/local/lib/pythonX.Y/dist-packages`
  - `/usr/lib/pythonX.Y`
  - `/usr/lib/pythonX.Y/lib-dynload`
  - `/home/user/.local/lib/pythonX.Y/site-packages`
  - `/usr/lib/python3/dist-packages`

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 110

110

## PEP 8 - Příkazy `import`

- Vždy na začátku souboru za docstring a komentáře, tj. před konstanty a globální proměnné
- Absolutní import – preferovaný protože čitelnější
  - ✓ `from some_module import some_class`
- Relativní import – akceptovatelné, ale nedoporučuji
  - ✗ `from .some_module import some_class`
  - ✗ `from .some_package import some_function`
  - ✗ `from . import some_class`

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 111

111

## PEP 8 – Pořadí příkazů `import`

- Každý modul/balíček na samostatné řádce (doporučuji stejně, je-li import více funkcí, objektů, atd.)
- Příkazy `import` rozděleny na sekce podle druhů modulů/balíčků
- Sekce oddělené prázdnou řádkou
- Popř. každá sekce abecedně řazená
- Speciální sekce – příkaz `future`: `from __future__ import ...`

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 112

112

## PEP 8 – Pořadí příkazů `import`

<b>PEP 236</b>	0 { • příkaz <code>future</code>	0
<b>PEP 8</b>	1 { • standardní knihovna	1
	2 { • dobře známé knihovny třetích stran	2
	3 { • ostatní knihovny třetích stran	3
	4 { • vlastní obecné knihovny	4
	5 { • moduly projektu	5

• Popř. v rámci sekcí abecedně **Doporučení**

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 113

113

## Druhy modulů a balíčků

- standardní knihovna
  - <https://docs.python.org/3/library/index.html>
- knihovny třetích stran
  - příkaz `pip install xxxxx`
  - příkaz `python -m pip install xxxxx`
  - příkaz `conda install xxxxx`
  - PyPi = Python Package Index ... <https://pypi.org/>
  - Anaconda repository ... <https://anaconda.org/anaconda/repo>

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 114

114



## Příkazy `import` (Google)

- Použití plných jmen pro moduly z balíčků
- Pozn. zastaralé rozlišování importu z balíčků a „pod-balíčků“
- Tj. **preferovat tečkovou notaci před `from ... import`**

ANO: `import matplotlib.pyplot`  
`import package2.subpackage1.module5` ✓

NE: `from matplotlib import pyplot` ✗  
`from package2.subpackage1 import module5` ✗

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 121

121

## Příkazy `import .. as` (Google)

- **Nepřejmenovávat při importu pomocí `as`**
- Výjimky:
  - Obvyklé zkratky pro dobře známé knihovny třetích stran
  - Opravdu dlouhá jména
  - Řešení nekompatibilit různých verzí

ANO: `import matplotlib.pyplot as plt`  
`import numpy as np` ✓

NE: `import utils.terminal_width as terminal_width` ✗

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 122

122

## Příkazy `import .. as` (Google)

- **Povolené výjimky** pro přejmenování importu: ✓

• <code>datetime</code>	<code>dt</code>
• <code>matplotlib.pyplot</code>	<code>plt</code>
• <code>multiprocessing</code>	<code>mp</code>
• <code>numpy</code>	<code>np</code>
• <code>pandas</code>	<code>pd</code>
• <code>seaborn</code>	<code>sns</code>
• <code>tensorflow</code>	<code>tf</code>
• <code>tkinter</code>	<code>tk</code>

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 123

123

## Souvislosti příkazu `import`

- PEP 20 - The Zen of Python

„Explicit is better than implicit.“

- ⇨ Nepoužívat `import` pomocí hvězdičky
- ⇨ Nepoužívat `import` jednotlivých tříd, funkcí, ...
- ⇨ Použití plných jmen pro importy modulů z balíčků
- ⇨ Nepřejmenovávat při importu pomocí `as`

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 124

124

## Struktura souboru s kódem (nejen PEP)

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 125

125

## Python struktura souboru

Shebang	<code>#!/usr/bin/env python3</code>
Kódování (nepovinné)	<code>## -*- coding: utf-8 -*-</code>
Docstring – dokumentace	<code>""" ... docstring modulu ... """</code>
Komentář k souboru (nepovinné)	<code>''' ... další popis ... '''</code>
Hlavička souboru (autor, atd.)	<code>_author_ = ...</code>
Import modulů a balíčků	<code>import ...</code>
Konstanty a globální proměnné	<code>DEBUG = ...</code>
Třídy a funkce	<code>def ... / class ...</code>
Spuštění programu (u balíčků nepovinné)	<code>if __name__ == "__main__": ...</code>

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 126


126

## Python shebang

- Povinné
- První řádek souboru
- Pro Linux/Unix/POSIX určuje interpret

ANO: `#!/usr/bin/env python3`

NE: `#!/usr/bin/python3`  
 NE: `#!/usr/bin/env python`  
 NE: `#!/usr/bin/python`



březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 127

127

## Python kódování souboru

- Volitelné
- Druhý řádek souboru
- Obecně: **UTF-8** (Python 3); ASCII (Python 2)
- Tj. netřeba: `# -*- coding: utf-8 -*-`

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 128

128

## Docstring modulu (souboru)

- Povinné
- Dokumentace vnořená v program
- PEP 257 - Docstring Conventions
- Tento text bude zobrazován jako nápověda v IDE, atd.
- Docstring: 3x uvozovky `""" text """` (!!!)

```
"""
text
"""
```

- Pro uživatele kódu (modulu, balíčku, knihovny, ...)

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 129

129

## Komentář na úrovni modulu

- Volitelné
- Nebude součástí nápovědy zobrazované v IDE
- Varianty:
  - Blokový komentář : `##` a mezera
  - Víceřádkový řetězec: 3x apostrof `''' text '''` (!!!)

```
##
text
'''
text
'''
```

- Pro editory/údržbě kódu (modulu, balíčku, knihovny, ...)

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 130

130

## Python struktura souboru

Shebang	<code>#!/usr/bin/env python3</code>
Kódování (nepovinné)	<code># -*- coding: utf-8 -*-</code>
Docstring – dokumentace	<code>""" ... docstring modulu ... """</code>
Komentář k souboru (nepovinné)	<code>''' ... další popis ... '''</code>
Hlavička souboru (autor, atd.)	<code>_author_ = ...</code>
Import modulů a balíčků	<code>import ...</code>
Konstanty a globální proměnné	<code>DEBUG = ...</code>
Třídy a funkce	<code>def ... / class ...</code>
Spuštění programu (u balíčků nepovinné)	<code>if __name__ == "__main__": ...</code>

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 131

131

## Python hlavička souboru

```

__author__ = "David Koniček"
__maintainer__ = "David Koniček"
__email__ = "david.konicek@cvut.cz"
__copyright__ = "\xa9 2022 CIIRC CTU, Prague."
                " All rights reserved."
__license__ = "MIT"
__version__ = "0.1.0"
__date__ = "2023/01/02"
__status__ = "Development"
__credits__ = []
__all__ = []

```

**textové řetězce**

**seznam řetězců**

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 132

132

## Python hlavička souboru

- Metadata programu/modulu
- „Nepsaný standard“ pro všechnu programátorskou práci, nejen Python
- Každý programovací jazyk má svoji syntaxi
- Python vychází z pravidel pro Docstring dle Epydoc  
<https://epydoc.sourceforge.net/manual-fields.html#module-metadata-variables>
- Python vychází z pravidel tvorbu balíčků PyPA (Python Packaging Association)  
<https://www.pypa.io/en/latest/>
- Rob Knight: Python Coding Guidelines (for Cogent project)  
[https://web.archive.org/web/20111010053227/http://jaynes.colorado.edu/PythonGuidelines.html#module\\_formatting](https://web.archive.org/web/20111010053227/http://jaynes.colorado.edu/PythonGuidelines.html#module_formatting)

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 133

133

## Python hlavička souboru

- `__author__` – řetězec (!!!), obsahuje jedno či více jmen (oddělovat čárkami nebo středníkem) ✓
- `__maintainer__` – řetězec (!!!), obsahuje jedno či více jmen (oddělovat čárkami nebo středníkem) 👍
- `__email__` – řetězec (!!!), obsahuje jedno či více emailových adres (oddělovat čárkami nebo středníkem) ✓

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 134

134

## Autorská práva a licence

- **Copyright = právo autora**, vzniká automaticky autorovi
- **Licence = právní ujednání**, definuje:
  1. práva užití a obvykle
  2. obvykle omezuje odpovědnost za škodu
- *What's the difference between Copyright and Licensing?*  
<https://opensource.stackexchange.com/questions/297/whats-the-difference-between-copyright-and-licensing>

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 135

135

## Autorská práva

- **Copyright = právo autora**; vzniká automaticky ze zákona
- Autorské dílo = nehmotný duševní výtvor, které musí mít znaky autorského díla (tvůrčí činnost; jedinečnost; zhmotněné)
- Legislativa: zákon 121/2000 Sb., popř. předpisy EU, popř. předpisy země užití, popř. celosvětové dohody
- Pozn.: slovo **copyright** a značka © jsou zvykově používány (do 1988 v USA závazně), ale v ČR/EU nemají právní význam
- Pro laiky je přijatelný zdroj informací česká Wikipedie  
[https://cs.wikipedia.org/wiki/Autorský\\_zákon\\_\(Česko,\\_2000\)](https://cs.wikipedia.org/wiki/Autorský_zákon_(Česko,_2000))

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 136

136

## Hlavička souboru: `__copyright__`

`__copyright__` 👍

- **Copyright = právo autora**; vzniká automaticky ze zákona  
⇒ tento text je deklaratorní, resp. informativní o držitelé práv
- Obvyklý text: držitel práv, rok/roky a deklarace  
`__copyright__ = "xa9 2022 CIIRC CTU, Prague." " All rights reserved."`

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 137

137

## Hlavička souboru: `__copyright__`


`__copyright__` – varianty možného textu: 👍

- 1) Student: dle ustanovení §60 zákona o školních dílech  
⇒ rok/roky, fyzický autor, jméno školy, popř. fakulty
- 2) Zaměstnanec: dle ustanovení §58 zákona o zaměstnaneckých dílech ⇒ rok/roky, zaměstnavatel
- 3) Speciálně uzavřené smlouvy – podle jejich textu
- 4) Ostatní případy ⇒ rok/roky, fyzický autor  
(Pozn. *nebylo konzultováno s právním oddělením*)

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 138

138

## Licence = právní ujednání




- Pro laiky je přijatelný zdroj informací a zejména **porovnáni** anglická Wikipedie: [https://en.wikipedia.org/wiki/Software\\_license](https://en.wikipedia.org/wiki/Software_license)
- **OSI Approved Licenses**: obsáhlý výčet licencí volně dostupného software (GNU, MIT, Apache, ...): <https://opensource.org/licenses/>
- Proprietární software přináší mnoho dalších typů licencí
- Praxe přinesla jiné texty pro software (GNU/GPL, MIT, APL, ...) a pro ostatní autorská díla (CC = Creative Commons, ...)

březen-duben '24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 139

139

## Licence = právní ujednání





- Doporučuji se řídit dle **OSI Approved Licenses** (mezinárodní) <https://opensource.org/licenses/?categories=international>
- Protože jsme členský stát EU (všech 23 jazyků, 27 zemí): <https://joinup.ec.europa.eu/collection/eupl/eupl-text-eupl-12>
- Doporučuji vybírat z licencí: **MIT, GPL, EUPL**
- Další možnost licence - WTFPL, ale pozor, neřeší zřeknutí se odpovědnosti za škody: <http://www.wtfpl.net/> <https://en.wikipedia.org/wiki/WTFPL>

březen-duben '24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 140

140

## Hlavička souboru: `__license__`




`__license__` – výběr textu: 


- Základ: váš právní status (student, zaměstnanec, zcela volný)
- Důrazně doporučuji neměnit text a držet se toho, co právníci vymysleli; text je vesměs univerzální ve smyslu různých národních úprav.
- Obvykle řešení:
  - Výběr jedné z licencí: **MIT / GPL / EUPL**
  - standardní zkratka podle webu OSI Approved Licenses
  - příložený textový soubor s názvem **LICENSE** nebo **LICENSE.TXT** s doslovným (!!!) zněním licence

březen-duben '24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 141

141

## Python hlavička souboru: `__date__`




`__date__ = "2024-04-03"` 


- volitelné
- datum poslední změny
- textový řetězec (!!!)
- formát podle ISO 8601, resp. ČSN ISO 8601  
⇒ YYYY-MM-DD
- ideálně automaticky doplňuje verzovací systém (GIT, atd.)

březen-duben '24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 142

142

## Hlavička souboru: `__status__`



• Status by měl odpovídat logice číslu verze (alfa, beta, release candidate, atd.) 

`__status__` – řetězec (!!!), obvykle jedno z

```
"Prototype"
"Development"
"Production"
```

březen-duben '24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 143

143

## Python číslování verzí



`__version__ = "5.8.4"` – řetězec (!!!) 

- PEP 440 - Version Identification <https://peps.python.org/pep-0440/>
- Inspirováno – sémantické verzování <https://semver.org/>
- Alternativa – kalendářové verzování <https://calver.org/>
- Reálný svět – marketing rozhoduje ⇒ nejrůznější kombinace a texty





březen-duben '24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 144

144



## Obvyklé sémantické verzování

- N.N.N .... číslo.číslo.číslo ... např. 1.43.15 ... jako řetězec
- [major],[minor],[release],[build]

Verze	Význam a použití
MAJOR	změně architektury nebo změna narušující zpětnou kompatibilitu
MINOR	přidání funkcionality při zachováním zpětné kompatibility
RELEASE / PATCH / MICRO	opravy chyb při zachováním zpětné kompatibility a žádné změny funkcionality
BUILD	neustále rostoucí číslo reprezentující vnitřní verzi (málo používané)

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 145

145

## Číslování verzí podle PEP 440

v	Epoch	1	Release	Optional	Pre-release	Optional	Pre-release	Optional	Dev	Optional	Local	Optional
[N!]	[N(.N)*]	[a b rc]	[3N]	[.postN]	[.devN]	[+.*]						

**N!** Epocha (nepovinné) = změna logiky číslování  
**N(.N)\*** Vydání [release] několik čísel, obvykle N.N.N  
**a|b|rcN** Předběžné verze [pre-release]  
**.postN** Dodatečná verze [post-release] (nepovinné)  
**.devN** Vývojové verze [development]

• PEP 440 ⇒ nová verze ⇒ PyPa Version specifiers  
<https://packaging.python.org/en/latest/specifications/version-specifiers>

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 146

146

## Číslování verzí podle PEP 440

**[N!][N(.N)\*][a|b|rc3N][.postN][.devN][+.\*]**

**release**      **development / internal / revision**

- .devN** • Vývojové verze, lze kombinovat s **a|b|rc|post|dev**
- aN** • Alfa verze = neobsahuje všechnu funkcionality
- bN** • Beta verze = veškerá funkcionality, netestováno
- rcN** • Release Candidate = interně testováno, „skoro“ dobré, probíhá externí testování
- finální (produkční) verze; tj. bez **a|b|rc|post|dev**

• Pozn.: *alfa - beta - RC - produkční* verze odpovídá standardním fázím vývojového cyklu software

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 147

147

## Python hlavička souboru

**\_\_credits\_\_** – seznam řetězců (!!!), **ne**obsahuje autory importovaných modulů, ale obsahuje autory m.j. 👍

- doporučení a rad
- kteří nahlásili chyby
- použitých fragmentů kódu
- inspirativních textů

```
__credits__ = ["Rob Knight", "Peter Maxwell"]
```

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 148

148

## Python hlavička souboru

**\_\_all\_\_** seznam (!!!) řetězců jmen tříd, funkcí, konstant, atd., které modul **exportuje** ✅

- Tj. jediná část metadat **používaná jazykem Python**
- Zároveň zvyšuje čtivost kódu

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 149

149

## Příkazy **import** a seznam **\_\_all\_\_**

**from xxx import \*** a importovaný modul **neobsahuje** **\_\_all\_\_** ⇒ importuje opravdu všechno ❌

**from xxx import \*** a importovaný modul **obsahuje** **\_\_all\_\_** ⇒ importuje jen vyjmenované v **\_\_all\_\_** ❌

**import xxx** a aktuální modul obsahuje použití objektů z **importovaného** modulu ⇒ kontroluje, že použité je v **\_\_all\_\_** ✅

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 150

150

## Python struktura souboru

Shebang	<code>#!/usr/bin/env python3</code>
Kódování (nepovinné)	<code># -*- coding: utf-8 -*-</code>
Docstring – dokumentace	<code>""" ... docstring modulu ... """</code>
Komentář k souboru (nepovinné)	<code>''' ... další popis ... '''</code>
Hlavička souboru (autor, atd.)	<code>_author_ = ...</code>
Import modulů a balíčků	<code>import ...</code>
Konstanty a globální proměnné	<code>DEBUG = ...</code>
Třídy a funkce	<code>def ... / class ...</code>
Spuštění programu (u balíčků nepovinné)	<code>if __name__ == "__main__": ...</code>

březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 151

151

## Import, konstanty, proměnné, třídy, funkce

- **Importy** – členění do sekcí podle PEP8
- **Konstanty** – názvy podle PEP8; popisné
- (Popř.) **globální proměnné** – názvy podle PEP8; popisné; preferovaně s podtržítkem na začátku
- **Třídy**
- **Funkce**

⇒ V celém modulu není žádný přímo vykonávaný kód

Python program v „klasickém“ pojetí

březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 152

152

## “Přímý kód” v souboru

- V celém modulu není žádný přímo vykonávaný kód
- Výjimky:
  - Příkazy import
  - Definice konstant
  - Inicializace globálních proměnných
  - Patička s případným voláním main()
- Další výjimky:
  - Testovací programy (unittest, pytest, nose2, ...)
  - Grafické programy (PyQT5, Tkinter, wxPython, ...)
  - Webové frameworky (Django, Flask, ...)

březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 153

153

## Patička souboru

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

- **main()** obvykle (variantně):
  - CLI aplikace
  - GUI aplikace
  - spuštění testů
  - spuštění démona
  - spuštění paralelních vláken / procesů

• What Does if \_\_name\_\_ == "\_\_main\_\_" Do in Python?  
<https://realpython.com/if-name-main-python/>

březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 154

154

## Struktury adresářů, projektů a balíčků

březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 155

155

## Projekt

- Obvyklé varianty výsledku:
  - Modul
  - Program, aplikace, skript
  - Knihovna, balíček
  - Spočtená data, grafy, atd.
- Obvykle ve formě repozitáře na (GIT) serveru
- Doporučené uspořádání adresářů balíčku

Project/

```

├── .gitignore
├── requirements.txt
├── README.md
├── LICENSE
├── pyproject.toml
├── setup.cfg
├── project/
│   ├── __init__.py
│   ├── main.py
│   ├── other code
│   └── tests/
│       ├── __init__.py
│       ├── test_main.py
│       └── ... other tests
├── doc/
├── scripts/
└── project
```

Změnit na jméno projektu

březen-duben 24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 156

156

## Distribuce projektů a balíčků

- Nástroje pro vytváření distribučních balíčků:
  - ~~distutils~~ ⇒ setuptools ⇒ build ??? Flit ??? Poetry ??? PyInstaller ?? Pyexe ??? Niutka ??? auto-py-to-exe
- setup.py/ install\_requires ... pip install XXX (instaluje používané moduly)
- pip.exe / python -m build / Poetry / Conda

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 157

157

## Distribuce projektů a balíčků

- Distribuční balíčky:
  - built distribution = bdist (ready-to-install):
    - egg (distribuce a balíčkování; instalace runtime) ⇔
    - wheels (distribuce a balíčkování) (\*.whl ~ \*.zip)
    - source distribution = sdist (\*.tar.gz)
- Konfigurační soubory pro distribuční balíčky:
  - setup.py + setup.cfg ⇔ pyproject.toml
- requirements.txt ... pip install -r requirements.txt (vývoj)

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 158

158

## Distribuce projektů a balíčků

- Při hledání informací, pozor na zastaralost údajů
- Python Packaging User Guide  
<https://packaging.python.org/en/latest/>
- PyPA (Python Packaging Authority)  
<https://www.pypa.io/en/latest/>
- setuptools (zdroj – PyPa)  
<https://setuptools.pypa.io/en/latest/index.html>
- build (zdroj – PyPa)  
<https://build.pypa.io/en/stable/index.html>
- Python Application Layouts: A Reference (zdroj – RealPython)  
<https://realpython.com/python-application-layouts/>

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 159

159

## Programátorské techniky (nejen) v Pythonu

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 160

160

## Programátorské techniky

- Clean code / human-friendly code / readable code / manageable code ⇔
- nikdo nenapíše napoprvé kód čistý (a obvykle ani funkční) ⇔
- refaktoring** (úprava kódu bez změny funkcionality) ⇔
- automatické testy (podporují kvalitu tím, že refaktoring nepřinese chyby)

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 161

161


## „Velká“ paradigmatata programování

- Strukturované programování = Python standardně ✓
- Imperativní programování = program = sekvence příkazů; akceptovatelní jen při testování, prototypování, atd. ✗
- Procedurální programování = program sekvence volání procedur 👍
- Funkční programování = funkce **map()**, **reduce()**, ..., uzávěry [closure], lambda, ... ✓
- (Deklarativní programování) = výjimečně

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 162

162

## „Velká“ paradigmata

- Objektové programování (OOP) = preferovaná technika 
- Modulární programování = preferovaná technika
- Návrhové vzory [Design patterns]
- Automatické testování
- CI/CD = continuous integration / continuous delivery

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 163

163



## Principy návrhu kódu

- Vesměs stručné principy s dalekosáhlými důsledky
- **KISS** = **Keep It Simple, Stupid** ⇨
  - (1) funkce/metody jsou malé
  - (2) srozumitelný kód bez „rafinovanosti“
- **YAGNI** = **You Aren't Going to Need It** ⇨
  - neprogramovat funkcionalitu, kterou nepotřebujeme a možná (!!!) budeme potřebovat v budoucnu
  - žádné programování „pro strýčka Příhodu“

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 164

164

## Principy návrhu kódu

- **DRY** = **Don't Repeat Yourself** ⇨
  - (1) podobný/stejný kód se neopakuje, ale je společnou funkcí/metodou/třídou
  - (2) osobní/týmová/firemní knihovna znovupoužitelného kódu
- **DIE** = **Duplication Is Evil** ... cca stejné
- **WET** = **Write Everything Twice** ... antipattern 

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 165

165

## Principy návrhu kódu

- **SESE** = **Single Entry, Single Exit** = one return only ⇨
  - funkce/metoda více obsahuje vnořené příkazy **if**
  - cykly častěji obsahují **break**
  - funkce má lokální proměnou pro výsledek
  - return je poslední příkaz
- **Robustness** principle = liberalismus pro vstupní hodnoty (akceptujte libovolný vstup; zpracujte přijatelný vstup), konzervatismus pro výstup (výsledky přesněm a v přesném formátu)

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 166

166

## Principy návrhu kódu (OOP)

- OOP přináší další principy návrhu kódu včetně **architektury**
- **SRP** = **Single Responsibility Principle** ⇨
  - funkce/metody/třídy dělají jen jednu jasně určenou věc ⇨
  - „vrstvení kódu“ do řady malých komponent
- **CQS** = **Command-Query Separation** ⇨ funkce/metoda:
  - (1) buď „něco počítá/dělá“
  - (2) nebo vrací hodnotu (stav, výsledek)
 Pozn. obvykle vyžaduje požití OOP

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 167

167


## Principy návrhu kódu (OOP)

- **SoC** = **Separation of Concernes** ⇨
  - členit kód do menších částí (funkce / třídy / moduly)
  - omezovat vzájemné vazby

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 168

168

## Principy návrhu kódu (OOP)



- **SOLID = SRP + OCP + LSP + ISP + DIP**
- **SRP** = Single responsibility principle = funkce/metody/třídy dělají jen jednu jasně určenou věc
- **OCP** = Open-closed principle = třída je otevřená k rozšíření / změně funkcionality (podtřídami; děděním), ale uzavřená pro nutnost změny základní třídy.
- **LSP** = Liskov substitution principle = odkazy a reference na podtřídy lze při použití nahradit (substituovat) odkazy/referencemi na základní třídy

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 169

169

## Principy návrhu kódu (OOP)





- **SOLID = SRP + OCP + LSP + ISP + DIP**
- **ISP** = Interface segregation = Interface třídy (metody, atd.) je oddělen od způsobu implementace. Třídy mají více malých oddělených rozhraní než jedno velké komplexní.
- **DIP** = Dependency inversion = třídy mají být závislé na rozhraní používaných tříd/funkcí, ne na jejich implementaci.

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 170

170

## Nevhodné techniky






- (antipatterns, smelling code)
- spaghetti code (dlouhé funkce/metody) ⇒ rozdělovat na funkce, délka cca 1÷2 obrazovky
- pizza code (příliš vnořených volání; příliš mnoho vrstev) ⇒ střídavě vytvářet „vrstvy“ kódu
- složitost / komplexita kódu ⇒ rozdělovat na funkce/metody, délka cca 1÷2 obrazovky; příliš vnořovat složené příkazy
- složitost / komplexita výpočtu ⇒ používat kvalitní algoritmy, používat ustálené knihovny, čerpat z kvalitní literatury

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 171

171

## Nevhodné techniky






- používání jmen bez informační hodnoty ⇒ identifikátory s jasnou sémantikou podle jména; dodržovat jmenné konvence
- ignorování zavedených „best practices“ ⇒ používat kvalitní algoritmy, používat ustálené knihovny, čerpat z kvalitní literatury
- konstanty jsou skryté ⇒ konstanty uvádět v konfigurační části kódu nebo v konfiguračním souboru
- „znovuvynalézání kola“ ⇒ používat kvalitní algoritmy, používat ustálené knihovny, čerpat z kvalitní literatury

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 172

172

## Nevhodné techniky



- nadměrné používání globálních proměnných ⇒ ideálně žádné globální proměnné; používat třídy
- zakomentovaný kód ⇒ nepoužívat, starý kód je ve verzovacím systému (GIT, ...)
- ladící kód stále aktivní či jen zakomentovaný ⇒ důsledně mazat nebo konstrukce `if DEBUG: ...`
- nejsou automatické testy ⇒ psát testy (`unittest`, `pytest`, `nose2`)

<https://realpython.com/python-testing/>

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 173

173

## Nevhodné techniky





- neudržované komentáře neodpovídající kódu ⇒ udržovat komentáře; používat komentáře v omezené míře
- nadměrné používání nesrozumitelných „one-liners“ ⇒ volání samostatných funkcí nebo použití složených příkazů
- copy/paste kódu bez porozumění ⇒ porozumět, testovat, popř. refaktoring
- AI vytvořený kód nekontrolovaný a bez porozumění ⇒ porozumět, testovat, popř. refaktoring

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 174

174

## Nevhodné techniky




- výjimky jsou sice zachycené, ale nezpracované/ignorované ⇒ zpracovávat výjimky, minimálně chybové hlášení (logování)
- výjimky jsou zachytávány všechny a bez rozlišení catch all ⇒ rozlišovat výjimky, použít web/AI/experimenty k nalezení vracených výjimek
- Ignorování návratových chybových stavů a výjimek ⇒ zpracovávat návratové kódy a výjimky
- používání/vytváření kódu a knihoven, které nehlásí chyby ⇒ zpracovávat a hlásit chybové stavy a výjimky

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 175

175

## Nevhodné techniky




- ... a další na individuální úrovni
- ... a mnoho dalších při týmové práci
- ... a mnoho dalších při řízení projektů

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 176

176

## Refaktoring




- **Refaktoring identifikátorů / názvů** s cílem lepších identifikátorů a/nebo redukce komentářů
- **Refaktoring struktury příkazů** s cílem jednoduššího a srozumitelnějšího kódu
- **Refaktoring výpočetní logiky** s cílem jednoduššího, srozumitelnějšího nebo rychlejšího kódu
- **Refaktoring struktury tříd, funkcí a metod** s cílem menších tříd/metod/tříd, vyšší srozumitelnosti ...

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 177

177

## Metody vývoje a řízení



- Projektové řízení
- Vývoj software
- Agilní metodiky řízení
  - Scrum
  - Extreme Programming (XP)
  - Kanban
  - ... a mnoho dalších
- Agilní týmy/organizace
  - adaptace agilních metodik do velkých firem / projektů
  - Spotify metodika (squads / chapters / tribes / guild)

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 178

178


## Programátorské techniky v Pythonu




březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 179

179

## Python techniky („Pythonish“)



- pojmenovávat konstanty, popř. \*.env
- type hinting
- výjimky jako podtypy standardních výjimek, nikoliv textové řetězce
- for count, item in enumerate (some\_list)
- řetězení řetězců pomocí "".join(), nikoliv pomocí +
- regulární výrazy, přehledný zápis

březen-duben '24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 180

180

## Python techniky („Pythonish“)

- typy Enum, Flag, ...
- sekvenční typy a jejich použití (list, tuple, UserList, namedtuple, kolekce, ...)
- typy pro pole/matice a jejich použití (list, tuple, dataframe, numpy array, numpy structured array)
- dekorátory funkcí, metod, tříd, ...
- Abstraktní datové typy collections.abc
- funkční programování
- Repräsentace obrázku

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 181

181

## Python techniky („Pythonish“)

- logging
- arg
- env
- I/O
- Closure
- Dekorátory
- Paralelní programování (vlákna, multiprocessing, asynchronní I/O, ...)

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 182

182

## Python OOP techniky („Pythonish“)

- OOP metodika
- veřejné, chráněné a privátní členy třídy
- dědění
- funkce super()
- @dataclass
- @classmethod
- @staticmethod
- @final
- @abstractmethod
- @property / @xxx.setter / @xxx.getter / @xxx.deleter
- magické metody („dunder“)

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 183

183

## PEP 20 - The Zen of Python

- Principy psaní „pythonic“ kódu
- Velmi stručné, velmi hutné, často obtížné dodržet

• <https://peps.python.org/pep-0020/>  
 • <https://pep20.org/>

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 184

184

## PEP 20 - The Zen of Python

- **Beautiful** is better than ugly.
- **Explicit** is better than implicit.
- **Simple** is better than complex.
- Complex is better than complicated.
- **Flat** is better than nested.

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 185

185


## PEP 20 - The Zen of Python

- **Sparse** is better than dense.
- **Readability** counts.
- Special cases are **not special enough to break the rules.**
- Although **practicality** beats purity.
- **Errors should never pass silently.**
- Unless explicitly silenced.

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 186

186

## PEP 20 - The Zen of Python




- In the face of ambiguity, refuse the temptation to guess.
- There should be one - and preferably only one - obvious way to do it.
- Although that way may not be obvious at first unless you're Dutch.
- Now is better than never.

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 187

187

## PEP 20 - The Zen of Python



- Although never is often better than "right" now.
- If the implementation is hard to explain, it's a bad idea.
- If the **implementation is easy to explain**, it may be a good idea.
- **Namespaces** are one honking great idea -- let's do more of those!

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 188

188


## Nástroje




březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 189

189

## Doporučené nástroje




- Tzv. statická analýza kódu, tj. bez spuštění kódu
- Původně – samostatné nástroje, obvykle CLI (**C**ommand **L**ine **I**nterface = příkazová řádka)
- Moderně – integrace do grafických IDE (**I**ntegrated **D**evelopment **E**nvironment)
- Použití umělé inteligence (AI = **A**rtificial **I**ntelligence)

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 190

190

## Python IDE




- **Visual Studio** Community/Professional/Enterprise (Microsoft)
- **Visual Studio Code** (Microsoft)
- **PyCharm** (JetBrains): Community/Professional
- **Jupyter**
- **Spyder**
- **PyDev**/LiClipse/Eclipse
- **PyScripter**
- Omezeně: editory (vim, Emacs, Atom, Sublime, Notepad++, ...)
- Omezeně: (Python) IDLE

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 191

191

## Výhody (Python) IDE



- Zvýraznění syntaxe
- Našeptávání/dokončování
- Ladění
- Statická analýza zobrazená v kódu
- Integrace GIT
- Integrace s AI
- Nástroje pro další jazyky a soubory (HTML, CSS, Markdown, make, shell, ...)

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 192

192



Příkaz	PyCharm	Visual Studio Code
Všechny příkazy	Ctrl + Shift + A	Ctrl + Shift + P
Formátování kódu	Ctrl + Alt + L	Shift + Alt + F
„Zakomentování“ kódu	Ctrl + /	Ctrl + / Ctrl+ háček
Našeptávač	Ctrl + mezera	Ctrl + mezera
Refaktoring – přejmenuj identifikátoru	Shift + F6	F2
Refaktoring – vytvoř proměnnou	Ctrl + Alt + V	???
Refaktoring – vytvoř funkci/metodu	Ctrl + Alt + M	???
Run / continue	F9	F5
Step into	F7	F11
Step over	F8	F10

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 193

193

## Nástroje umělé inteligence

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 194

194

## Proč (učit) na ČVUT nástroje AI?

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 195

195

## Základní problém s AI

- Otázky legality, autorství, plagiátorství, opisování ⇒

1. Vaše morální integrita
2. Firmy atd: ochrana duševního vlastnictví, ochrana dat, ochrana osobních dat, ...
3. Legislativa

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 196

196

## AI - otázky legality a autorství

⇒ metodické pokyny ČVUT

- Metodický pokyn č. 5/2023 - Rámcová pravidla používání umělé inteligence na ČVUT pro studijní a pedagogické účely v Bc a NM studiu  
<https://www.cvut.cz/legislativa-tykajici-se-studia#umela-inteligence>
- Metodický pokyn č. 2/2024 o dodržování etických principů při přípravě vysokoškolských závěrečných prací (zejména články 2.2, 2.5, 2.6)  
<https://www.cvut.cz/legislativa-tykajici-se-studia#eticke-principy>

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 197

197

## AI - otázky legality a autorství

...

- 2.2 nepřivlastňovat si cizí myšlenky a nápady či výsledky výzkumu bez uvedení jejich původního zdroje (nedopouštět se plagiátorství),
- ...
- 2.5 důsledně citovat autory, jejichž myšlenky nebo texty byly použity ve vlastní práci (při použití cizího díla ve větším rozsahu je vhodné si vyžádat i svolení autora),
- 2.6 uvést použití výstupů nástrojů umělé inteligence (dále jen UI),
- ...

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 198

198

## Vlastnosti nástrojů AI

- Metody NLP, GPT, vesměs statistické textové modely !!!
- ⇨
- Někdy chybné až nesmyslné výsledky („halucinace“).
- Nejasná autorská práva.
- Riziko krádeže obsahu (při dotazech a při integraci s IDE).
- Kvalita výsledku dána kvalitou vstupů a trénování modelů. Obvykle neznáte ani jedno.
- Kvalita výsledků dále dána postprocessingem (filtrací). Obvykle neznáte.

**ALE**

- Neustálý a bouřlivý rozvoj.

březen-duben'24 David Koníček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 199

199

## Doporučení k použití nástrojů AI

1. Používat !
2. Zohlednit pravidla ČVUT.
3. Přiznávat použití (použité zdroje, vygenerování kódu, ...).
4. Kvalifikační práce – doporučuji vést si podrobné záznamy o využití AI.
5. Nepoužívejte na kód a data, které nesmí opustit „počítač“.

březen-duben'24 David Koníček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 200

200

## Doporučení k použití nástrojů AI

6. Nepoužívejte žádný výsledek AI, kterému nerozumíte.
7. Konzervatismus až nedůvěra v kvalitu výsledků z nástrojů umělé inteligence.
8. Kontrola správnosti odpovědí / výsledků.
9. Úprava podle vlastního úsudku a/nebo potřeb.
10. Popř. použít jen jako radu, inspiraci, atd., ale nepoužívejte doslovně (tj. nepožívat „copy – paste“).

- Pozn. kód z GPT často používá příliš krátké identifikátory.

březen-duben'24 David Koníček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 201

201

## AI nástroje pro programování

- Pozn.: názvy technologií versus obchodní názvy produktů
- Pozn.: časté změny názvů z marketingových důvodů
- Velké množství
- Použití/generování:
  - Obecné texty
  - Obrázky
  - Zvuky/hudba
  - Videá
  - Specializované texty, zejména programování

březen-duben'24 David Koníček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 202

202

## AI nástroje pro programování

- **OpenAI GPT-3.5**/GPT-4
- **MS Copilot**
- **GitHub CopilotX**, zdarma akademická licence
- **Tabnine** (0-12-39 USD/u/m)
- Amazon CodeWhisperer (0-19). Meta Code Llama,
- **Codeium** (free, \$15/u/m)
- Google Duet AI for Developers = Gemini Code Assist = Bard (\$19/u/m)
- GPT4All – open-source, offline
- tabby – open-source, offline

březen-duben'24 David Koníček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 203

203

## Microsoft Copilot

- Copilot Windows 11 ≠ Copilot 365 ≠ CopilotX = GitHub Copilot
- Využívá GPT-4
- Windows 11 Copilot (preview, rollout Sep23, zdarma)
  - vyžaduje Windows 11, 23H2, pravděpodobně též vyžaduje verzi EN a Ent/EDU
- Copilot web (<https://copilot.microsoft.com/>) / Bing search
  - GPT: Copilot/Designer/Vacation planner/Cooking ass./Fitnes trainer
- (GitHub) Copilot (X) ... možnost akademické licence

březen-duben'24 David Koníček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 204

204

## Microsoft Copilot – další verze

- Copilot for Microsoft 365 (30\$/u/m, min 300u, Office 365 vč. Teams)
- Copilot for Sales (2024)
- Copilot for Service (2024)
- Copilot in Viva
- Microsoft Security Copilot

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 205

205

## Práce s AI nástroji

- Prompt = výzva = zadání pro práci
- Systém prompt
- API
- Context window

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 206

206

## Možnosti AI při programování

- „Chytrý“ vyhledávač (náhrada Google, DuckDuckGo, Bing, ...)
- Dotazy v přirozeném jazyce
- Našeptávání [autocompletion]
- Generování kódu
- Generování testů
- Refaktoring kódu

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 207

207





## Možnosti AI při programování

- Detekce a opravování chyb v kódu
- Detekce chyb v bezpečnosti
- Tvorba dokumentace (Docstring i běžné komentáře)
- Vysvětlení kódu
- Type hinting
- Překlady jazyků (rodný jazyk ⇔ English)
- Vylepšení jazykových formulací („Czenglish“ ⇔ English)

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 208

208

## Způsoby použití AI

-  Dotaz / odpověď na webu
-  Dotaz / odpověď na webu v IDE
-  Našeptávače IDE
-  Generování kódu, komentářů, atd. v IDE

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 209

209


## AI – 1. dotaz/odpověď na webu

- Obecná webová rozhraní systémů GTP
  - Např: OpenAI GTP-3.5/GPT-4, MS Copilot, GitHub CopilotX, Meta Code Llama
  - Princip dotaz – odpověď, obvykle s udržení kontextu mezi dotazy
  - Zásadní „trik“ – nastavení režimu práce tzv. promptem/výzvou

březen-duben'24 David Koniček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 210

210

## AI – 2. dotaz/odpověď v IDE




⇒ AI má plný přístup k Vašemu kódu a lze očekávat, že si ho stáhne a použije ke svému učení

- Obvykle vyžaduje instalaci doplňků (*pozn. situace jaro 2024*)
- Integrovaná webová rozhraní v IDE
  - Obvykle již přednastavený vhodný prompt (tj. režim práce)
  - Obvykle jazykové modely trénované jen nad relevantními texty
  - Integrace s editačním oknem

březen-duben'24 David Koníček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 211

211

## AI – 3. našeptávače




⇒ AI má plný přístup k Vašemu kódu a lze očekávat, že si ho stáhne a použije ke svému učení

- Obvykle nabízí IDE, též možnost jako instalaci doplňků (*pozn. situace jaro 2024*)
- Našeptávače (on-line navrhování kódu)
  - Dříve – podle pravidel jazyka a signatur tříd/metod/funkcí, resp. statisticky jaká slova byla použita v jiných programech (*Tabnine*)
  - Nově – návrhy i delších fragmentů kódu včetně použití správných již existujících proměnných

březen-duben'24 David Koníček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 212

212

## AI – 4. generování kódu v IDE



⇒ AI má plný přístup k Vašemu kódu a lze očekávat, že si ho stáhne a použije ke svému učení

- Obvykle nabízí IDE, též možnost jako instalaci doplňků (*pozn. situace jaro 2024*)
- Integrované nástroje v IDE (nativně nebo plugin)
  - generování kódu
  - generování Docstring
  - generování typehint
  - reformulace angličtiny
  - atd.

březen-duben'24 David Koníček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 213

213


## Další nástroje



březen-duben'24 David Koníček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 214

214

## Nástroje kontroly kódu




- Moderní speciální nástroje – **SonarQube / SonarLint**
- Statická kontrola kódu v IDE
- Klasické nástroje kontroly kódu – přístup přes IDE
- Klasické nástroje kontroly kódu – přístup přes CLI

březen-duben'24 David Koníček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 215

215

## Nástroje kontroly kódu



- PEP8 formátování / styl → **pycodestyle/pep8, Flake8**
- Datové typy (*type checker*) → **MyPy, Pyright, Pytype**
- Detekce chyby (*error linter*) → **PyLint, pyflakes, Flake8**
- Nepoužitý / mrtvý kód → **Vulture, eradicate**
- Komplexita / nepřehlednost → **McCabe, Radon**
- Bezpečnost (hlavně web) → **Bezpečnost: Bandit**

Nečistý kód (*code smells*)

březen-duben'24 David Koníček - jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 216

216

## Další nástroje

- Balíčkování (*packaging*): Pyroma
- Formátování kódu podle PEP8: black, autopep8
- Formátování Docstring: pydocstringformatter, docformatter

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 217

217

## Doporučené nástroje – pep8

- pycodestyle (= pep8) kontrola stylu podle PEP 8
- Integrovan v PyCharm
- Dokumentace: <https://pycodestyle.pycqa.org/en/latest/intro.html>
- Seznam pravidel/chyb – vhodné pro konfiguraci výjimek: <https://pycodestyle.pycqa.org/en/latest/intro.html#error-codes>  
<https://pypi.org/project/pep8-naming/>

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 218

218

## Doporučené nástroje – flake8

- flake8 (= PyFlakes + pycodestyle + McCabe)
  - Pycodestyle = pep8 = kontrola stylu
  - Pyflakes = kontrola syntaxe a hrubých chyb programování
  - McCabe = kontrola cyklomatické complexity
- Dokumentace <https://media.readthedocs.org/pdf/flake8/latest/flake8.pdf>
- Seznam pravidel/chyb – vhodné pro konfiguraci výjimek: <https://flake8.pycqa.org/en/latest/user/error-codes.html>  
<https://www.flake8rules.com/>

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 219

219

## Doporučené nástroje

- Pylint
- MyPy
- SonarLint
- pydocstyle
- PyCharm & VisualStudioCode – nutné instalovat

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 220

220

## Kvalita kódu

- **(Statická) složitost / komplexita** kódu jako míra pochopitelnosti, přehlednosti, udržovatelnosti ...

*versus*

- **Výpočetní složitost / komplexita** / náročnost – čas, paměť, O(n)

*versus*

- Složitost použitých (teoretických) metod, tj. matematika, fyzika, ....

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 221

221


## Statická komplexita

- **Měření složitosti** s cílem získání metriky **pro zlepšování kódu**:
  - Cyklomatická komplexita
  - Počet řádek kódu
  - Počet řádek vykonatelného kódu
  - Halsteadovy index(y), speciálně Halsteadův objem
  - Index udržovatelnosti – kombinace výše uvedených
  - Kognitivní komplexita
- **Vysoká komplexita (obvykle) správně indikuje nepřehledný kód.**

březen-duben'24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 222

222

## Statická komplexita




- **Cyklostatická komplexita** (Thomas McCabe, 1976. DoD)
  - Cca počítá definice funkcí/metod, větvení a cykly a jejich vnořování
- **Halsteadův objem** (Maurice Howard Halstead, 1977)
  - Cca počítá složitost výrazů a proměnných
- **Index udržitelnosti** – kombinace výše uvedených plus počtu řádků

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 223

223

## Komplexita / nepřehlednost




- **Kognitivní komplexita** (G. Ann Campbell, 2016?, SonarSource)
  - Penalizuje:
    - narušení lineárního toku algoritmu (shora dolů, zprava doleva)
    - vnořené struktury
    - Tj. cca: `if`, `match`, `for`, `while`, `try-catch`, `break`, `continue`, rekurze, lambda funkce, složité logické výrazy, ternární operátory
  - Nepenalizuje:
    - definice funkcí, metod a tříd
  - Lépe vyhovuje moderním programovacím jazykům a moderním paradigmatům.

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 224

224

## Řešení příliš vysoké komplexity




- (Refaktoring identifikátorů / názvů)
- **Refaktoring struktury příkazů**:
  - jednodušší podmínky (tj. vyčlenit výpočet podmínky do samostatného příkazu)
  - inverze podmínky
  - vnořené `if` ⇒ řetězec příkazů `if ... elif ... elif ...`
  - vnořené `if` nebo řetězec `if ... elif ... elif ...` ⇒ příkaz `match`

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 225

225

## Řešení příliš vysoké komplexity





- **Refaktoring struktury funkcí a metod**:
  - vnořené cykly a podmínky ⇒ rozdělit na více menších funkcí/metod
  - dlouhý kód ⇒ rozdělit na více menších funkcí/metod/modulů
  - funkce/metody řízené (větvené) podle hodnot logických parametrů (tzv. příznaků) ⇒ použít dědění tříd; použít volání funkcí podle hodnoty příznaku
  - „včasné“ příkazy `return`, je ale v rozporu s principem SESE
  - zredukovat duplicitní kód (viz princip DRY) ⇒ společný kód do parametrizované společné funkce/metody

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 226

226

## Zdroje informací a sebevzdělávání

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 227

227

## ... zdarma



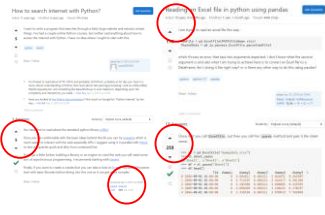
- <https://realpython.com/>
- <https://towardsdatascience.com/>
- <https://www.geeksforgeeks.org/>
- <https://www.digitalocean.com/community/>
- <https://datagy.io/>
- <https://docs.python.org/3/tutorial/index.html>
- <https://www.khanacademy.org/computing>
- <https://pypi.org/> .... najít balíček a k němu dokumentaci
- <https://www.edx.org>

březen-duben'24 David Koniček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 228

228

## ... zadarmo

- Weby významných univerzit:  
MIT, Stanford, Harvard,  
Cambridge, Berkeley ...
- <https://stackoverflow.com/>  
... klasický zdroj informací  
pro programátory, pozor na  
kvalitu, pozor na stáří textu



březen-duben '24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 229

229

## ... zadarmo nebo za málo

- <https://knihy.nic.cz/>
- <https://greenteapress.com/wp/think-python/>
- <https://github.com/pamoroso/free-python-books>
- <https://pythonbooks.org/free-books/>
- **Humble Bundle** – e-knihy renomovaných vydavatelství s  
výraznou slevou  
<https://www.humblebundle.com/>



březen-duben '24 David Koníček - Jak programovat v Pythonu (ČVUT FEL B3B33LAT - Laboratoře robotiky) 230

230