1. Úvod do programování

BAB37ZPR – Základy programování

Stanislav Vítek

Katedra radioelektroniky
Fakulta elektrotechnická
České vysoké učení v Praze

Přehled témat

Část I

O předmětu

I. O předmětu

Předmět a lidé

- Webové stránky předmětu

  https://cw.fel.cvut.cz/wiki/courses/bab37zpr
- Přednášející a garant předmětu
  - Stanislav Vítek, viteks@fel.cvut.cz

    http://mmtg.fel.cvut.cz/personal/vitek/
- Cvičící
  - Matěj Oravec, oravemat@fel.cvut.cz
  - Václav Vencovský, vencovac@fel.cvut.cz
- Konzultace
  - MS Teams – po domluvě

Cíle předmětu

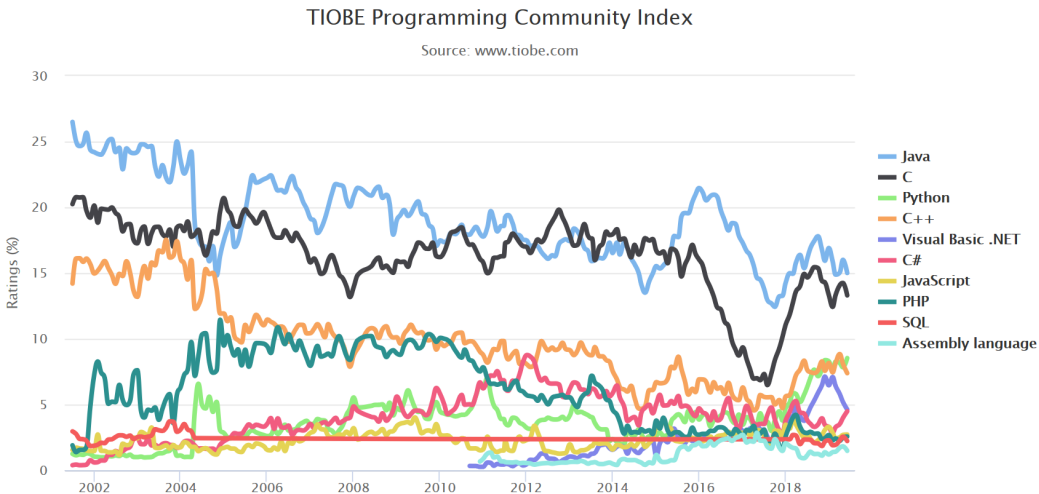
- **Motivovat k programování**
  - Programování je klíčová dovednost, která může hrát rozhodující roli na trhu práce
- **Naučit se algoritmizovat**
  - Formulace problému a návrh řešení
  - Rozklad problému na dílčí úlohy
  - Identifikace opakujících se vzorů
- **Získat zkušenosti s programováním**
  - Základní programovací konstrukce

    Proměnné, cykly, podmínky, datové struktury a jednodušší algoritmy
  - Programovací jazyk Python, řada principů obecně použitelných

    Cvičení, domácí úkoly, hledání chyb, práce s dokumentací, test

Programátorovi nestačí perfektní znalost programovacího jazyka, ale především musí vědět, jak vůbec danou úlohu řešit.

Proč Python?


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
Java
C
Python
C++
Visual Basic .NET
C#
JavaScript
PHP
SQL
Assembly language

Proč Python? 2/2

- jazyk vysoké úrovně, všeobecné použití, dobře čitelný
- velmi populární, mnoho knihoven, multiparadigmatický
- dynamický, interpretovaný (byte-code)
- s automatickou alokací paměti

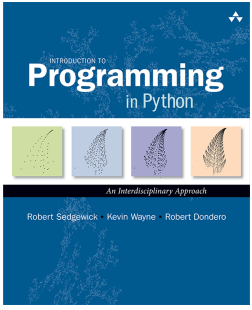
Organizace a hodnocení předmětu

- **Studijní výsledky**
  - Průběžná práce v semestru – domácí úkoly a test
  - Zkouškový a implementační test
- **Docházka**
  - Přednášky jsou nepovinné, ale snad přínosné a zábavné
  - Cvičení jsou povinná, možné dvě omluvené absence

    V případě distanční výuky není vyžadována online účast, k dispozici bude audiovizuální záznam cvičení.
  - Na cvičení se očekává aktivní účast při řešení příkladů

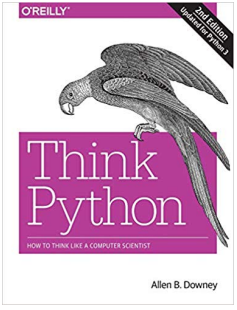
    Na cvičení je třeba se **připravit**, nejlépe návštěvou přednášky a studiem podkladů (řešené příklady)
- **Řešení problémů**
  - Obracejte se na svého cvičícího
  - Při komunikaci e-mailem pište vždy ze své fakultní adresy
  - Do předmětu zprávy uvádějte zkratku předmětu ZPR
  - V případě zásadních problémů uvádějte do CC též přednášejícího

Zdroje a literatura



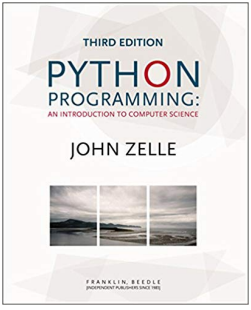
Robert Sedgewick
Introduction to Programming in Python: An Interdisciplinary Approach
Addison-Wesley
2015

ISBN 978-0134076430



Allen B. Downey
Think Python: How to Think Like a Computer Scientist
O'Reilly Media
2015

ISBN 978-1491939369



John Zelle
Python Programming: An Introductio
Computer Science
Franklin, Beedle & Associates
2016

ISBN 978-1590282755

I. O předmětu

Domácí úkoly

- Samostatná práce s cílem osvojit si praktické zkušenosti s programováním
- Jednotné zadání na prednášce a jednotný termín odevzdání
- Náročnost domácích úkolů se postupně zvyšuje
- Odevzdání domácích úkolů prostřednictvím systému BRUTE
- Cílem řešení úkolů je získat **vlastní** zkušenost
  - Neopisujte – škodíte především sobě
  - Provádíme automatickou kontrolu plagiátů u všech odevzdaných řešení
    - každý s každým
    - každý s řešením z minulých let (pokud je podobný příklad)
    - u podezřelých případů provedeme manuální kontrolu
  - V případě odhalení jsou potrestání **oba** účastníci incidentu

Pokud nečemu nerozumíte, ptejte se!

Hodnocení

Zdroj bodů	Maximum	Nutné minimum
Domácí úkoly	30	15
Test v semestru	20	10
Semestrální práce	20	10
Zkouška	30	15
Součet	100	50

- Za práci v semestru je třeba získat nejméně 35 bodů.
- Z 6 domácích úloh je třeba odevzdat 4 dle vlastního výběru. Úkoly mají stanovené deadlines, pozdní odevzdání je penalizováno. Domácí úkoly musí být odevzdány nejpozději do 09.1.2022 ve 23:59 CET!
- Test v semestru – test na počítači, teoretické otázky (zveřejněny na CW)
- Zkouška – implementace několika příkladů na počítači, cca 4 hodiny

Klasifikace

Klasifikace	Bodové rozmezí	Slovní hodnocení
A	≥ 90	výborně
B	$80 - 89$	velmi dobře
C	$70 - 79$	dobře
D	$60 - 69$	uspokojivě
E	$50 - 59$	dostatečně
F	< 50	nedostatečně

Přehled přednášek

1.	Informace o předmětu, úvod do programování. Primitivní datové typy.	20.9.
2.	Řidící struktury, větvení a cykly. Funkce	4.10.
3.	Pole, seznam, řetězec. Vyhledávání a řazení.	11.10.
4.	Principy objektově orientovaného programování.	18.10.
5.	Algoritmy pro práci čísly. Konečný automat.	25.10.
6.	Záznam, fronta, zásobník.	1.11.
7.	Spojové seznamy, stromy. Rekurze.	8.11.
8.	Zvolnění – asocitivní pole a množina, rozptylová tabulka.	15.11.
9.	Aplikace – live coding, feedback.	22.11.
10.	Aplikace – numerické výpočty, vizualizace dat.	29.11.
11.	Aplikace – zpracování signálu.	6.12.
12.	Aplikace – zpracování dat, machine learning.	13.12.
13.	Aplikace – Python v embedded technologiích.	20.12.
14.	Zkouškový test – předtermín.	3.1.

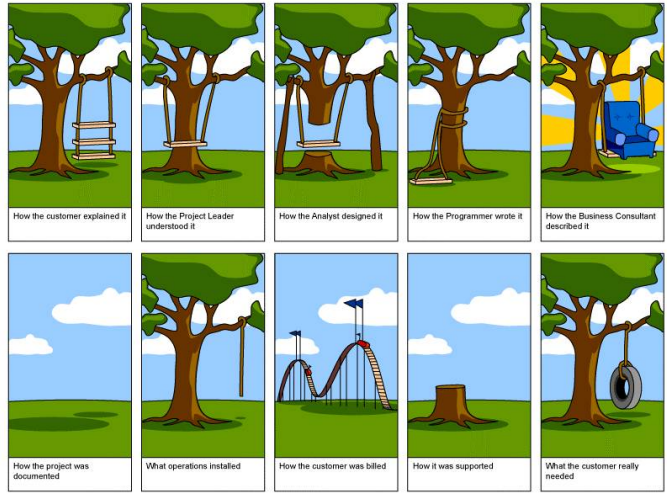
Část II

O programování

II. O programování

Řešení problémů

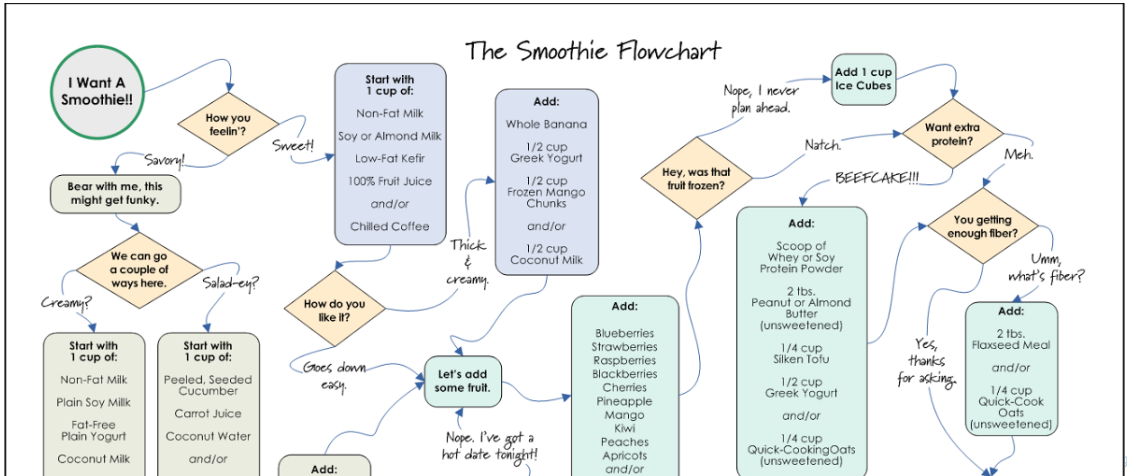
1. formulace problému
2. analýza možných řešení
3. návrh algoritmu
4. implementace
5. ověření funkčnosti
6. optimalizace
7. oprava chyb
8. údržba
9. dokumentace



Co je to program?

- Program je **recept** – posloupnost kroků (výpočtů), popisující průběh řešení nějakého problému pomocí dostupných prostředků – programovací prostředí, počítač, ...

Receptu budeme říkat algoritmus.



Co je to algoritmus?

- Návod nebo postup, jak provést určitou činnost.
- Algoritmus by měl být tak podrobný, aby mu porozumněl i počítač.
- Vlastnosti algoritmu:
  1. Skládá se z konečného počtu jednoduchých činností – kroků.
  2. Po každém kroku lze určit, jak se má pokračovat nebo skončit.
  3. Počet opakování jednotlivých kroků algoritmu je vždy konečný.
  4. Vede ke správnému výsledku.
  5. Algoritmus lze použít k řešení celé (velké) skupiny podobných úloh.

Manželka: kup chleba a když budou mít rohlíky, vezmi jich deset.
Manžel, programátor: přinese z obchodu deset chlebů, protože rohlíky měli.

https://www.youtube.com/watch?v=Ct-l0OUqmyY

Slovo algoritmus vzniklo odvozením od jména perského matematika Al-Chorezmího, jehož jméno bylo ve středověku latinizováno jako Al-Gorizmí.

Zápis algoritmu

- Existují 4 hlavní způsoby, jakými lze algoritmus popsat:
  - slovně – vyjádříme slovně postup řešení a jednotlivé kroky
  - graficky – použití vývojových diagramů a struktogramů
  - matematicky – jednoznačnný popis matematickou konstrukcí (např. rovnicí)
  - programem – kroky algoritmu jsou popsány programovacím jazykem
- Návrhy algoritmů:
  - shora dolů – problém rozdělíme na několik podúloh, které řešíme
  - zdola nahoru – z triviálních úloh skládáme vyšší úlohy
  - kombinace obou metod

> V praxi vždy záleží především na komplexnosti a povaze řešeného algoritmus, který postup bude nejlepší aplikovat.

Základní složky programů a algoritmů

- Programy zpravidla transformuje množinu vstupních dat na množinu dat výstupních
- Základní složky programů

  Vstup dat – načtení dat programem, interaktivní nebo ze souboru dat
  Popis dat – volba datového typu a umístění v paměti
  Zpracování – výpočet definovaný algoritmem, řízení toku programu
  Výstup – interakce s uživatelem nebo např. zápis do souboru

- Řízení toku programu

  Posloupnost – jeden nebo několik kroků, které se provedou právě jednou v daném pořadí
  Cyklus – opakování nějaké posloupnosti, dokud je splněna podmínka opakování
  Větvení – volba posloupnosti instrukcí na základě vyhodnocení podmínky

- Kombinace základních složek algoritmu umožňuje vytvářet komplexní konstrukce.
- Pokud se některé části algoritmu opakují, je vhodné poslupnosti orgranizovat do větších celků: procedur a funkcí (podprogramů).

Jak začít?

Jednoduché algoritmy, grafické programování

- Scratch – MIT Media Lab
- Angry Birds
- Code with Anna and Elsa
- Minecraft

Programovací jazyk Karel

- Pohyb robota po čtvercové sítí
- Richard E. Pattis, Karel The Robot: A Gentle Introduction to the Art of Programming, Stanford, 1981
- Online: Stanford , Oldřich Jedlička

Další zajímavé programovací jazyky pro výuku programování např. zde .

A co Python?

- Pokud máme Python nainstalovaný, stačí ho spustit

```
$ python3
Python 3.4.3 (default, Oct 14 2015, 20:28:29)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license"
for more information.
>>>
```

- Pokud Python nainstalovaný nemáme, tak si ho nainstalujeme :-)
  - `https://www.python.org/downloads`
    - včetně editoru IDLE
    - instalační nástroj `pip`
  - `https://www.anaconda.com/`
    - včetně IDE Spyder
    - instalační nástroj `conda` a správa prostředí pomocí Anaconda Navigator

II. O programování

Python jako kalkulačka

- Python lze pohodlně využívat v interaktivním módu

```
$ python3
Python 3.4.3 (default, Oct 14 2015, 20:28:29)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license"
for more information.
>>> 3+8
11
>>> 11*(5+3)
88
>>> 128./16.
8.0
>>> 2**16
65536
```

Python v interaktivním módu

- Píšeme **výrazy**, které obsahují
  - Celá čísla: `3`, `8`, ...
  - Reálná čísla: `128.`, `11.5`,...
  - Operátory: `+`, `-`, `/`, `*`,
  - Oddělovače: `(`, `)`
- Co se děje v zákulisí?
  - Spustili jsme program `python3`, **interpret** Pythonu
  - Opakované vykonávání (smyčka, *loop*)
    - tisk výzvy *(prompt)* `>>>`
    - přečtení uživatelského vstupu *(read)*
    - vyhodnocení výrazu *(evaluate)*
    - tisk výsledku *(print)*
- Poznámka k syntaxi
  - Python je **case sensitive** – velikost písmen je důležitá
  - Diakritika – Python3 umožňuje používat UTF kódování, raději to ale dělat nebudeme
  - Komentáře – symbol `#`

II. O programování

Proměnné a přiřazení

- Hodnotu výrazu lze uložit pro pozdější použití:

```
>>> a = 3    # identifikátor = výraz
>>> b = 3 + a
```

- Jaká je hodnota proměnné **b**?

```
>>> b
6
```

- Proměnná udržuje hodnotu, která se může měnit (proto proměnná)
- Proměnná má datový typ:
  - primitivní: číslo (int, float), pravdivostní hodnota (bool)
  - strukturovaný: řetězec (string), seznam / pole, slovník
- Názvy proměnných
  - posloupnost čísel, číslic a znaků '_', názvy by měly jasně vysvětlovat, jakou hodnotu popisují
  - víceslovné názvy proměnných: **nazvy_s_podtrzitky**, **CamelCase** a **mixedCase**.
  - nelze používat klíčová slova Pythonu (`if`, `else`, `True`, ...)

Proměnné – příklad 1

```
>>> boys=15
>>> girls=17
>>> total=boys+girls
>>> difference=girls-boys
>>> ratio=boys/total
>>> total
32
>>> difference
2
>>> ratio
0.46875
```

Proměnné – příklad 2

- Hodnoty proměnných lze měnit

```
>>> a=10
>>> a=a-2
>>> a=a*2
```

- Jaká je hodnota a?

```
>>> a
16
```

- Proč používat proměnné
  - **DRY** = Do not repeat yourself
  - Šetřme si práci, neopakujme se
  - Zlepšení
    - **Srozumitelnosti** – smysluplná jména proměnných
    - **Údržby** – jedna změna jen na jednom místě
    - **Efektivity** – využijeme předchozích výpočtů

Výrazy a operace

- výrazy – kombinace proměnných, konstant a volání funkcí pomocí operátorů
- operace – aritmetické, logické, řetězení textových řetezců, ...
- pořadí vyhodnocování se řídí prioritou a asociativitou

Příklad výrazy a operátory

```
x = 13
y = x % 4 # modulo
y = y + 1
y += 1
a = (x==3) and (y==2)
b = a or not a
s = "petr"
t = "klic"
u = s + t
```

Přiřazení a rovnost

- přiřazení `=`

  `x = 3` znamená "přiřaď do `x` hodnotu 3"
- test na rovnost `==`

  `x == 3` znamená "otestuj zda v `x` je hodnota 3"
- častá chyba: záměna `=` a `==`
- pozor: `is`

  `x is 3` se chová zdánlivě jako `==`, ale jsou tam rozdíly

  pokročilý programovací prvek pro porovnávání immutable (neměnných) objektů, teď nebudeme nepoužívat

Operátory

- Většina operátorů intuitivních
  - aritmetické: `*`, `/`, `+`, ...
  - logické: `and`, `or`
- Mírné odlišnosti od jiných jazyků
  - celočíselné dělení: `//`
  - mocnina: `**`
- Zkrácený zápis operátoru: `y += 5` odpovídá `y = y + 5`
- Pořadí vyhodnocování vesměs intuitivní (algebra)
- Pokud jste na pochybách
  - konzultujte dokumentaci
  - závorkujte
- Zkrácené vyhodnocení: `1 + 1 == 2 or x == 3`

II. O programování

První program

- vytvoříme v textovém editoru
- uložíme do souboru `hello.py`
- spustíme (z příkazové řádky, opakovaně)

```python
# Vypise pozdraveni
print("Hello world")
```

lec01/hello.py

```
$ python3 hello.py
Hello world
```

Příklad – převod jednotek teploty 1/4

- Kolik stupňů Celsia je 75 stupňů Fahrenheita?

```
>>> f=75
>>> c=(f-32)*5./9.
>>> print(c)
23.88888888888889
```

- Trochu hezčí výpis (pro pokročilé):

```
>>> print(f, " °F je ", c, " °C.")
75 °F je 23.88888888888889 °C.
```

- Další zlepšení:

```
>>> print("%f °F je %f °C." % (f, c))
75.000000 °F je 23.888889 °C.
>>> print("%0.1f °F je %0.1f °C." % (f, c))
75.0 °F je 23.9 °C.
```

Příklad – převod jednotek teploty 2/4

- Funkce `print` vytiskne své argumenty
- Argumentem funkce `print` může být číslo nebo řetězec
- `%f` do řetězce doplní reálná čísla z dalších argumentů
- Počet desetinných míst reálného čísla lze omezit

Co když chceme převést více hodnot?

- Šetřme si práci, neopakujme se
- **DRY** = Do not repeat yourself
- Vytvoříme program, který budeme moci opakovaně spouštět

Příklad – převod jednotek teploty 3/4

- V textovém editoru vytvoříme soubor `units.py`

```
1 # Prevod stupnu Fahrenheita na stupne Celsia
2 f=75
3 c=(f-32)*5./9.
4 print("%0.1f F je %0.1f C." % (f, c))
```

lec01/units.py

```
$ python units.py
75.0 F je 23.9 C.
```

- Program převádí pouze jednu hodnotu.
- Co třeba převádět hodnotu načtenou z příkazové řádky?

Příklad – převod jednotek teploty 4/4

- Vylepšená verze s načítáním čísla z příkazového řádku

```
1 # Prevod stupnu Fahrenheita na stupne Celsia
2 import sys
4 f=int(sys.argv[1]) # první argument
5 c=(f-32)*5./9.
6 print("%0.1f F je %0.1f C." % (f, c))
```

lec01/units2.py

```
$ python units.py 75
75.0 F je 23.9 C.
$ python units.py 60
60.0 F je 15.6 C.
$ python units.py -20
-20.0 F je -28.9 C.
```

II. O programování

Datové typy v programovacích jazycích

- Jak jsou typy deklarovány?
  - **explicitně** – zápis programátorem v kódu, např. `int x;`
  - **implicitně** – typ je určen automaticky kompilátorem
- Jak se provádí typová kontrola?
  - **staticky** – na základě kódu (při kompilaci)
  - **dynamicky** – za běhu programu

A co Python?

- Dynamické implicitní typování – typ se určuje automaticky a může se měnit
- **deklarace** proměnné – první přiřazení hodnoty
- zjištění typu: `type`, `isinstance`
- možnost explicitního přetypování

```
>>> x = float(y)
>>> y = str(158)
```

Datové typy v Pythonu

- Číselné – `int`, `float`, `complex`

```
a = 5
print(a, "je typ", type(a))

a = 2.0
print(a, "je typ", type(a))

a = 1+2j
print(a, "je komplexni cislo?", isinstance(a, complex))
```

- Seznam – `list`

```
a = [1, 2.2, 'python']
```

- Textový řetězec – `str`
- Další typy: uspořádaná n-tice `tuple`, množina `set`, slovník `dict`

II. O programování

Standardní výstup 1/3

- výpis proměnné (s automatickým odřádkováním)

```
>>> x = 10
>>> print(x)
10
```

- výpis s doprovodnou informací

```
# standardni zpusob, formatovani pomoci funkci
print('x = ', x)
# spojovani (concatenate) textovych retezcu
print('x = ' + str(x))
# Python 2.x style, formatovani pomoci modifikatoru
print('x = %d' % x)
# f-funkce, Python 3.6 a novejsi
print(f'x = {x}')
```

- tisk více hodnot

```
x = 10; y = 11; z = 12
print(x, y, z) # hodnoty automaticky oddeleny ' '
print(x, y, z, sep=';') # potlaceni separatoru ''
print(x, y, z, sep='\n') # odradkovani separatorem
print('x={}, y={}'.format(x,y))
print('x={1}, y={0}'.format(x,z))
```

```
10 11 12
10;11;12
10
11
12
x=10, y=11
x=12, y=10
```

Standadrní výstup 3/3

- bez odřádkování

```
print('Prvni', 'Druhy', 'Treti', sep=', ', end=', ')
print('Ctvrty', 'Paty', 'Sesty', sep=', ')
```

```
Prvni, Druhy, Treti, Ctvrty, Paty, Sesty
```

- formátovací funkce

  `.rjust`, `.ljust`, `.center`
  `.rstrip`, `.lstrip`

- další možnosti:

  `sys.write()`, ...

Příklad – formátovaný výstup 1/3

```
# %[flags][width][.precision]type
# integer a float
print("A : % 2d, B : % 5.2f" % (1, 05.333))
# integer
print("A : % 3d, B : % 2d" % (240, 120))
# octal
print("% 7.3o"% (25))
# float v exponencialnim tvaru
print("% 10.3E"% (356.08977))
```

```
A :  1, B : 5.33
A : 240, B : 120
031
3.561E+02
```

Tento způsob formátování pravděpodobně v další verzi Pythonu nebude.

Příklad – formátovaný výstup 2/3

```
# kombinace pozicniho argumentu a klicoveho slova
print('Na hristi jsou {0}, {1}, a {other}.'
  .format('Cesi', 'Slovaci', other ='rozhodci'))
# formatovani cisel
print('A :{0:2d}, B :{1:8.2f}'
  .format(12, 00.546))
# zmena pozicniho argumentu
print('B: {1:3d}, A: {0:7.2f}'.format(47.42, 11))
print('A: {a:5d},  B: {p:8.2f}'
  .format(a = 453, p = 59.058))
```

```
Na hristi jsou Cesi, Slovaci, a rozhodci.
A :12, B :    0.55
B:  11, A:   47.42
A:   453,  B:    59.06
```

Příklad – formátovaný výstup 3/3

```
str = "ahoj"
# tisk centrovaneho retezce a vyplnoveho znaku
print (str.center(40, '#'))
# tisk retezce zarovnaneho vlevo a vyplnoveho znaku
print (str.ljust(40, '-'))
# tisk retezce zarovnaneho vpravo a vyplnoveho znaku
print (str.rjust(40, '+'))
```

```
##################ahoj##################
ahoj--------------------------------------
++++++++++++++++++++++++++++++++++++ahoj
```