

Opakování, praktické tipy, programovací jazyky

IB111 Základy programování
Radek Pelánek

2018

- shrnutí základů, zkouškový Kahoot
- vývojová prostředí, editory
- správa verzí
- knihovny, praktický vývoj v Pythonu
- přehled programovacích jazyků
- návaznosti

- shrnutí základních témat
- co je potřeba hladce umět pro další studium
- upozornění na problematické body

Řízení toku výpočtu

- podmíněný příkaz
- cykly: for, while
- funkce, return

Datové typy

- čísla (int, float)
- řetězce
- seznamy, n-tice
- slovníky

Indexování

- indexování od nuly
- rozlišování mezi:
 - indexem i
 - hodnotou na příslušné pozici $a[i]$
- rozdíl mezi indexováním seznamu a slovníku
- vnořené struktury, indexování
 - $\text{data}[x][y]$ vs. $\text{data}[x,y]$
 - interpretace indexování „zleva“

- základní řadicí algoritmy: bubble sort, select sort, insert sort
- použití vestavěného řazení
 - `sorted(alist)` vs. `alist.sort()`
 - řazení podle kritéria: `key`

- dekompozice problému na funkce
- čistá funkce, vedlejší efekt
- předávání parametrů, změna parametrů
- rozlišování, vhodné použití:
 - funkce, která mění parametry
 - funkce, která vrací nový objekt

- globální, lokální
- vytvoření aliasu vs. kopie (mělká, hluboká)
- měnitelné vs. neměnitelné typy
 - indexování slovníku
 - předávání parametrů funkcím

- základní princip sebe-reference
- čtení, interpretace rekurzivního kódu
- jednoduchá rekurze s návratovou hodnotou

- rozlišení „třída“ vs. „objekt“
- metody, datové atributy
- vestavěné typy jsou objekty, použití objektové notace
- vlastní definice jednoduchých tříd, použití objektů
 - `__init__`
 - metody, význam `self`
 - datové atributy

Čtení komplexních výrazů

- výrazy podle priorit operátorů
- volání funkcí „zevnitř“
- indexování, tečková notace „zleva“

```
len(alice.mother.daughters[0].daughters)
```

```
data[x][get_value(a, b[3])]
```

```
putpixel(get_coordinates(n),  
         change_color(getpixel((x,y))))
```

Programátorský styl

- základní představa, co je (není) „pěkný“ kód
- PEP8 doporučení
- komentáře, dokumentace

IDLE dostatečný pro jednoduché příklady, do budoucna chcete něco lepšího. . .

žádoucí vlastnosti editoru:

- syntax highlighting
- odsazování, párování závorek
- autocomplete, suggest
- PEP8 kontrola
- podpora ladění
- podpora refaktORIZACE
- . . .

příklady různých typů editorů:

- **IDLE** základní editor používaný v tomto kurzu
- **emacs, vi** (+příkazová řádka) obecné editory, příp. se specifickou konfigurací
- **pyCharm** „silný“ editor speciálně pro Python, vhodné obzvláště pro velké projekty
- **ipython, jupyter** interaktivní použití (prolínání programu a výsledků), v prohlížeči

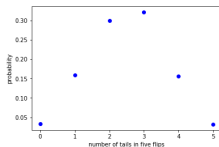
```
In [14]: rnd.seed(33)
dicethrow = rnd.randint(1, 6 + 1, 100)
side = np.zeros(6, dtype='int')
for i in range(6):
    side[i] = np.count_nonzero(dicethrow == i + 1)
    print('number of times', i + 1, 'is', side[i])
print('total number of throws ', sum(side))

number of times 1 is 17
number of times 2 is 17
number of times 3 is 15
number of times 4 is 24
number of times 5 is 19
number of times 6 is 8
total number of throws 100
```

[Back to Exercise 1](#)

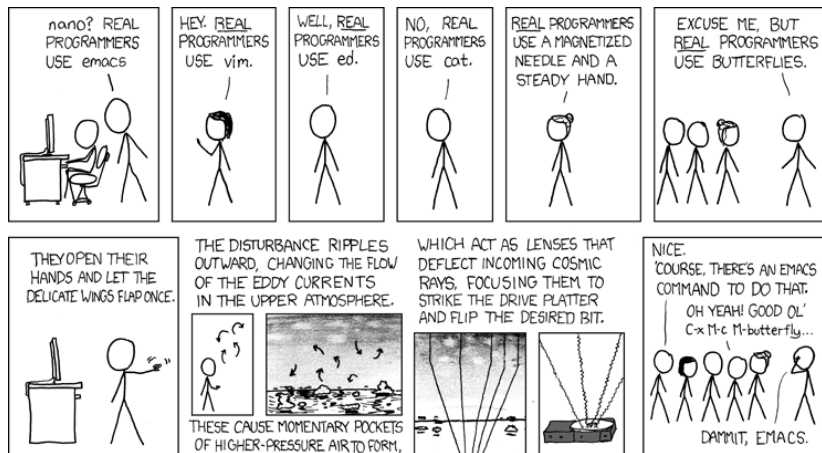
[Answers to Exercise 2](#)

```
In [15]: N = 1000
tails = np.sum(rnd.randint(0, 1 + 1, (5, 1000)), axis=0)
counttails = np.zeros(6)
for i in range(6):
    counttails[i] = np.count_nonzero(tails == i)
plt.plot(range(0, 6), counttails / N, 'bo')
plt.xlabel('number of tails in five flips')
plt.ylabel('probability');
```



http://nbviewer.jupyter.org/github/mbakker7/exploratory_computing_with_python/blob/master/notebook9_discrete_random_variables/py_exploratory_comp_9_sol.ipynb

xkcd: Real Programmers



<https://xkcd.com/378/>

- naivní přístup:
 - `myprogram.py`, `myprogram2.py`,
`myprogram_oct_24.py`
 - `myprogram_zaloha.py`, `myprogram_pokus.py`
 - `myprogram_final.py`, `myprogram_really_final.py`
- sofistikovanější přístup – „version control“
 - automatizovaná správa verzí
 - podpora týmové práce
 - mnoho různých řešení: `git`, `cvs`, `svn`, ...

- současné populární řešení
- distributed revision control
- GitHub – repositář, (primárně) veřejné projekty
- gitlab.fi.muni.cz – repositář na FI, umožňuje snadno vytvářet soukromé projekty

další témata pro praktické použití:

- funkcionální prvky: map, filter, lambda funkce, ...
- obecnější předávání parametrů, indexování, využití objektů, práce se soubory, ...
- iterátory, výjimky, ...
- využívání modulů

užitečné moduly v základní distribuci

- **math**: matematické funkce
- **random**: náhodná čísla
- **sys**: „systémové“ funkce a proměnné
- **os**: spolupráce s operačním systémem
- **re**: regulární výrazy
- **datetime**: práce s časem
- **json**: práce se soubory ve formátu JSON

Příklady známých „externích“ knihoven:

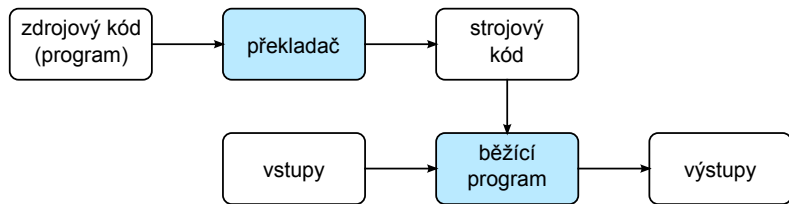
- **Django**: webový framework
- **NumPy, SymPy, SciPy**: efektivní numerické výpočty, symbolické výpočty, vědecké výpočty, statistika
- **Pandas**: práce s daty (především „tabulkovými“), SQL-like operace
- **matplotlib**: tvorba grafů, vizualizace
- **pygame**: vývoj her
- **scrapy**: „scrapování“ dat z webu
- **Tensorflow**: strojové učení, deep learning

Programovací jazyky

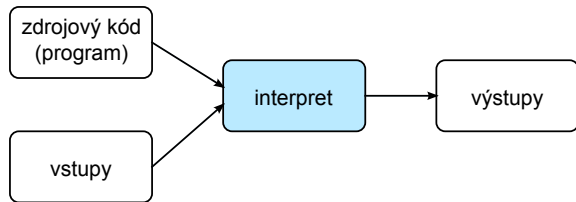
- přehled
- klasifikace, vlastnosti
- historie
- způsoby užití
- neseriózní postřehy

Interpretace, kompilace

kompilovaný program



interpretovaný program



Programovací jazyky: klasifikace I

nízko-úrovňové

- kompilované
- nutnost řešit specifika konkrétního systému
- explicitní práce s pamětí
- náročnější vývoj (nízká efektivita práce)
- vysoká efektivita programu

vysoko-úrovňové

- interpretované
- nezávislé na konkrétním systému
- využití abstraktních datových typů
- snadnější vývoj (vysoká efektivita práce)
- nižší efektivita programu

nikoliv dvě kategorie, ale plynulý přechod; zjednodušeno

Programovací jazyky: klasifikace II

zjednodušená klasifikace a použití

nízko-úrovňové C, FORTRAN, ...

vestavěné systémy, rychlé výpočty

objektové C++, Java, C#, ...

klasické aplikace, rozsáhlé systémy

skriptovací Python, PHP, JavaScript, Perl, ...

programování pro web, skriptování, prototypy

deklarativní Prolog, LISP, Haskell, ...

umělá inteligence

Jazyky se liší v práci s typy

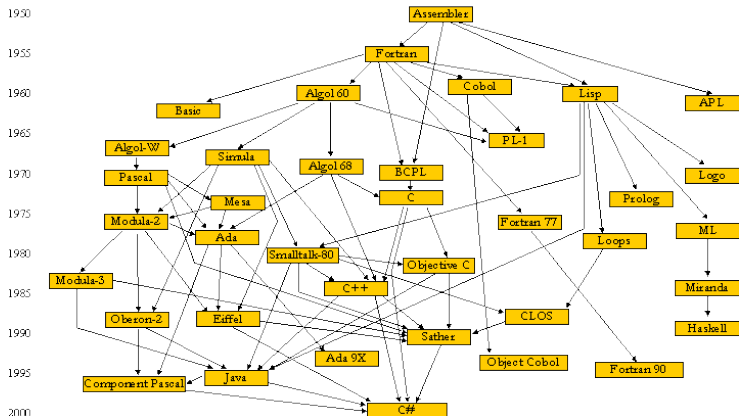
MY NEW LANGUAGE IS GREAT, BUT IT
HAS A FEW QUIRKS REGARDING TYPE:

```
[1] > 2 + "2"  
=> "4"  
[2] > "2" + []  
=> "[2]"  
[3] > (2/0)  
=> NaN  
[4] > (2/0)+2  
=> NaN  
[5] > "" + ""  
=> '+'  
[6] > [1,2,3]+2  
=> FALSE  
[7] > [1,2,3]+4  
=> TRUE  
[8] > 2/(2-(3/2+1/2))  
=> NaN.00000000000000013
```

```
[9] > RANGE(" ")  
=> (' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ')  
[10] > + 2  
=> 12  
[11] > 2+2  
=> DONE  
[14] > RANGE(1, 5)  
=> (1, 4, 3, 4, 5)  
[13] > FLOOR(10.5)  
=> |  
=> |  
=> |  
=> |___10.5___
```

<https://xkcd.com/1537/>

Programming Language Family Tree



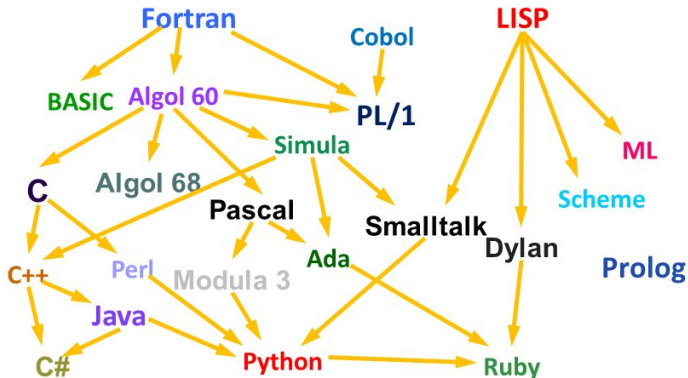
1 November, 2000

(C) Fachhochschule Aargau für
Technik, Wirtschaft und Gestaltung
Nordwestschweiz

10

A family tree of languages

Some of the 2400 + programming languages



Historie programovacích jazyků

prapočátky:

- 19. století: Charles Babbage, Ada Lovelace, děrné štítky, ...
- 30. léta: teoretické základy programování, Turingův stroj, lambda kalkul (Alonzo Church)
- 40. léta: první počítače, strojový kód, assembler

50. a 60. léta: první vysokoúrovňové jazyky (v některých aplikacích přežívají dodnes)

- ALGOL
- COBOL
- FORTRAN – vědecko-technické výpočty (užíván stále)
- BASIC

- „jazyk pro začátečníky“
- Beginner's All-purpose Symbolic Instruction Code
- rozšířen v 70. a 80. letech na „mikropočítačích“
- *výborný jazyk pro vytvoření špatných programátorských návyků*

BASIC

```
10 INPUT "What is your name: "; U$
20 PRINT "Hello "; U$
30 INPUT "How many stars do you want: "; N
40 S$ = ""
50 FOR I = 1 TO N
60 S$ = S$ + "*"
70 NEXT I
80 PRINT S$
90 INPUT "Do you want more stars? "; A$
100 IF LEN(A$) = 0 THEN GOTO 90
110 A$ = LEFT$(A$, 1)
120 IF A$ = "Y" OR A$ = "y" THEN GOTO 30
130 PRINT "Goodbye "; U$
140 END
```

<https://en.wikipedia.org/wiki/BASIC>

Historie programovacích jazyků

- 70. léta
 - rozvoj základních paradigmat (imperativní, objektové, funkcionální, logické)
 - C, Pascal, Prolog
- 80. a 90. léta
 - další rozvoj jazyků, specializace, nové prvky související např. s nástupem internetu
 - C++, Perl, Haskell, Ruby, R, Java, JavaScript, PHP
- současnost
 - nové verze jazyků
 - vznik nových jazyků: Go, Dart, Kotlin, Julia

Rosseta Code

- rosettacode.org
- stejné problémy řešené v mnoha programovacích jazycích



If programming languages were ...

- weapons

<https://9gag.com/gag/anXEbe0/if-programming-languages-were-weapons>

- religions

<http://blog.aegisub.org/2008/12/if-programming-languages-were-religions.html>

- boats

<http://compsci.ca/blog/if-a-programming-language-was-a-boat/>

- vehicles

http://crashworks.org/if_programming_languages_were_vehicles/

killing a dragon:

<https://blogs.oracle.com/roumen/how-to-kill-a-dragon-with-various-programming-languages>

Přehled programovacích jazyků

důležitý aspekt přehledu: různé jazyky mají různé rysy,
(ne)výhody a aplikační domény

nedůležitý aspekt přehledu: volba citátů a přirovnání (značně
subjektivní, pro zpestření, ...)

C would be Judaism – it's old and restrictive, but most of the world is familiar with its laws and respects them. The catch is, you can't convert into it – you're either into it from the start, or you will think that it's insanity. Also, when things go wrong, many people are willing to blame the problems of the world on it.

—

C is a nuclear submarine. The instructions are probably in a foreign language, but all of the hardware itself is optimized for performance.

- nízkoúrovňové programování
- „blízko hardwaru“
- optimalizace rychlosti výpočtu

mnoho jazyků staví na syntaxi C

základní rozdíly oproti Pythonu:

- vyznačování bloků kódu, (ne)významnost bílých znaků
- explicitně typovaný jazyk

C syntax: ukázka ciferný součet

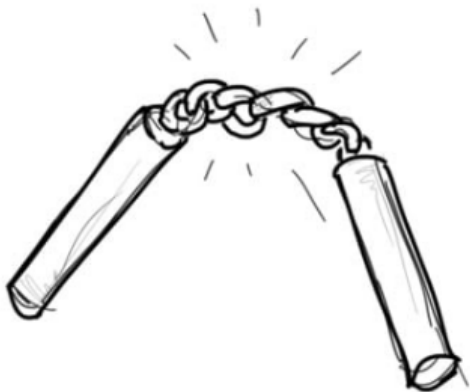
```
#include <stdio.h>
int SumDigits(unsigned long long n,
              const int base) {
    int sum = 0;
    for (; n; n /= base)
        sum += n % base;
    return sum;
}
int main() {
    printf("%d %d %d\n",
          SumDigits(1, 10),
          SumDigits(12345, 10),
          SumDigits(123045, 10));
    return 0;
}
```

Objektové jazyky odvozené od C

C++, C#, Java, ...

- kompilované (Java – bytecode)
- (většinou) explicitně typované
- typicky „silně objektové“
- vhodné pro „velké projekty“

C++ is a set of nunchuks, powerful and impressive when wielded but takes many years of pain to master and often you probably wish you were using something else.



Java is a cargo ship. It's very bulky. It's very enterprise~y. Though it can also carry a lot of weight. Will carry a project, but not very fun to drive.



Java: Hello World

```
public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello world!");
    }
}
```

Interpretované, „skriptovací“ jazyky

- JavaScript
- Python
- Perl
- PHP
- Ruby

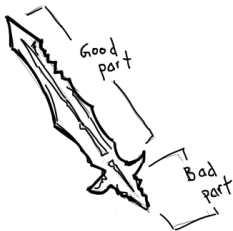
typické užití: vývoj webu (front-end, back-end), zpracování dat, skriptování, prototypování, ...

interpretované jazyky flexibilnější než kompilované

typická ukázka: příkaz `eval`

- „vyhodnocení výrazu v řetězci“
- může usnadnit práci
- ale nebezpečné (zejména nad uživatelským vstupem)

JavaScript is a sword without a hilt.



- i přes podobnost názvu nemá s Javou mnoho společného
- interpretovaný jazyk
- „jazyk webového front-endu“

PHP is a bamboo raft. A series of hacks held together by string. Still keeps afloat though.



Ruby

Ruby is difficult to describe. It's sleek, sexy, and very fun to drive. Here's a picture. Very trendy.



Perl

Perl would be Voodoo – An incomprehensible series of arcane incantations that involve the blood of goats and permanently corrupt your soul. Often used when your boss requires you to do an urgent task at 21:00 on friday night.

—

Perl used to serve the same purpose as Python, but now only bearded ex-hippies use it.



Deklarativní jazyky

- imperativní programování: program je posloupnost instrukcí („jak“ má počítač počítat)
- deklarativní programování: program je popis toho, co se má udělat

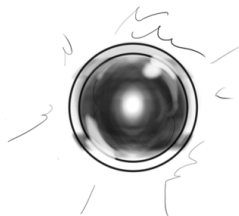
Deklarativní jazyky

- logické programování: Prolog
- funkcionální programování: Lisp, Haskell

typické užití: umělá inteligence, výpočty, „výuka principů, které využijete jinde“ (funkcionální prvky dnes v mnoha dalších jazycích)

Prolog

Prolog is an AI weapon, you tell it what to do, which it does but then it also builds some terminators to go back in time and kill your mom.



Prolog

```
mother_child(trude, sally).
```

```
father_child(tom, sally).
```

```
father_child(tom, erica).
```

```
father_child(mike, tom).
```

```
sibling(X, Y) :- parent_child(Z, X), parent_child(Z, Y)
```

```
parent_child(X, Y) :- father_child(X, Y).
```

```
parent_child(X, Y) :- mother_child(X, Y).
```


Lisp is a shiv which comes in many forms. Anyone who uses this is probably crazy and dangerous.



Lisp: ideální jazyk pro milovníky závorek.

```
(defun factorial (n)
  (if (= n 0) 1
      (* n (factorial (- n 1)))))
```



<https://xkcd.com/1312/>

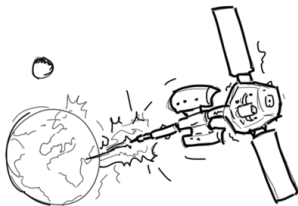
```
bsort :: Ord a => [a] -> [a]
bsort s = case _bsort s of
    t | t == s    -> t
      | otherwise -> bsort t
where _bsort (x:x2:xs) | x > x2    = x2:(_bsort (x:xs))
                       | otherwise = x:(_bsort (x2:xs))
      _bsort s = s
```

nástroje vyvinuté primárně jako „matematický software“, ale obsahují obecný programovací jazyk

- Mathematica
- MATLAB
- R

Mathematica

Mathematica is a low earth orbit projectile cannon, it could probably do amazing things if only anyone could actually afford one.



Jaký jazyk je nejlepší?

Jaký jazyk je nejlepší?

- neexistuje „univerzální“ jazyk, každý má své (ne)výhody
- lepší otázka: „Jaký jazyk je nejlepší pro danou situaci?“
 - problém, který řešíme
 - tým, který problém řeší
 - „legacy code“
- učte se různé jazyky

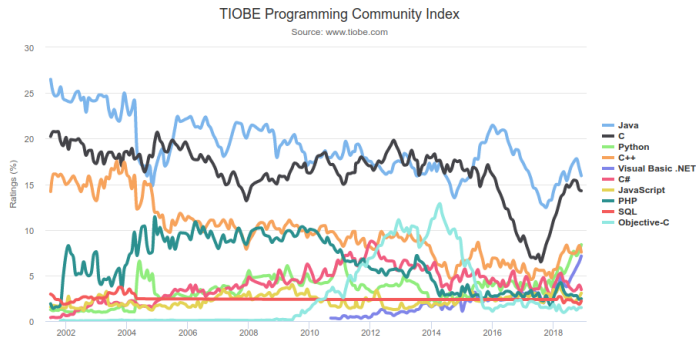
těžko měřitelné, existuje řada pokusů různými metrikami:

- vyhledávání na webu
- počty knížek o jazyku
- výskyty v inzerátech
- dotazy na StackOverflow (a podobných stránkách)
- projekty na GitHubu (a podobných repozitářích)

Popularita jazyků

na vršku se vesměs vyskytují (abecedně): **C, C++, Java, JavaScript, Python**

konkrétní příklad indexu popularity:



Co dál?

- algoritmy (směr k větší abstrakci)
 - **IB002** Algoritmy a datové struktury I – programátorské úlohy v Pythonu
 - navazující **IV003** Algoritmy a datové struktury II
- programování podrobněji (směr k nižší abstrakci, jak funguje počítač)
 - **PB071** Principy nízkoúrovňového programování – jazyk C, správa paměti, práce s řetězci
- objektově-orientované programování
 - **PB161** Programování v jazyce C++ (a navazující)
 - **PB162** Programování v jazyce Java (a navazující)

Co dál?

- jiná paradigmatata
 - **IB015** Neimperativní programování – Haskell, Prolog
- více o jazyce Python a jiných jazycích
 - **PV248** Kurz jazyka Python
 - **PV249** Vývoj v jazyce Ruby
 - **PV178** Úvod do vývoje v C#/.NET (a navazující předměty)
- teorie programování, vyčíslitelnost a složitost – učí se v rámci různých předmětů (IB102, IB107)

Závěrečné přání

Ať vás programování baví!