Praktické tipy, programovací jazyky

IB111 Základy programování
Radek Pelánek

2017

Dnešní přednáška

- shrnutí základů
- vývojová prostředí, editory
- správa verzí
- knihovny, praktický vývoj v Pythonu
- regulární výrazy
- přehled programovacích jazyků
- návaznosti
- Kahootová anketa

IB111 shrnutí

- 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
  - `data[x][y]` vs. `data[x,y]`
  - interpretace indexování „zleva“

Řazení

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

Funkce

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

Proměnné

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

Rekurze

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

Objekty

- 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))))
```

Vývojová prostředí, editory

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
- ...

Vývojová prostředí, editory

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 obzvlášť pro velké projekty
- **ipython, jupyter** interaktivní použití (prolínání programu a výsledků), v prohlížeči

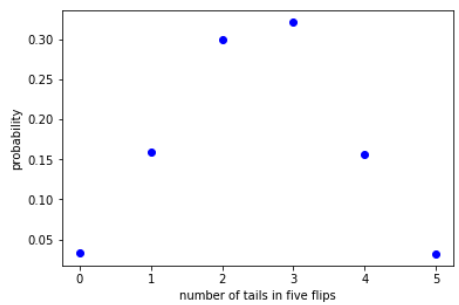
In [14]:
```
rnd.seed(33)
dicethrow = rnd.randint(1, 6 + 1, 100)
side = np.zeros(6, dtype='int')
for i in range(6):
    side[i] = np.count_nonzero(dicethrow == i + 1)
    print('number of times', i + 1, 'is', side[i])
print('total number of throws ', sum(side))
```

```
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

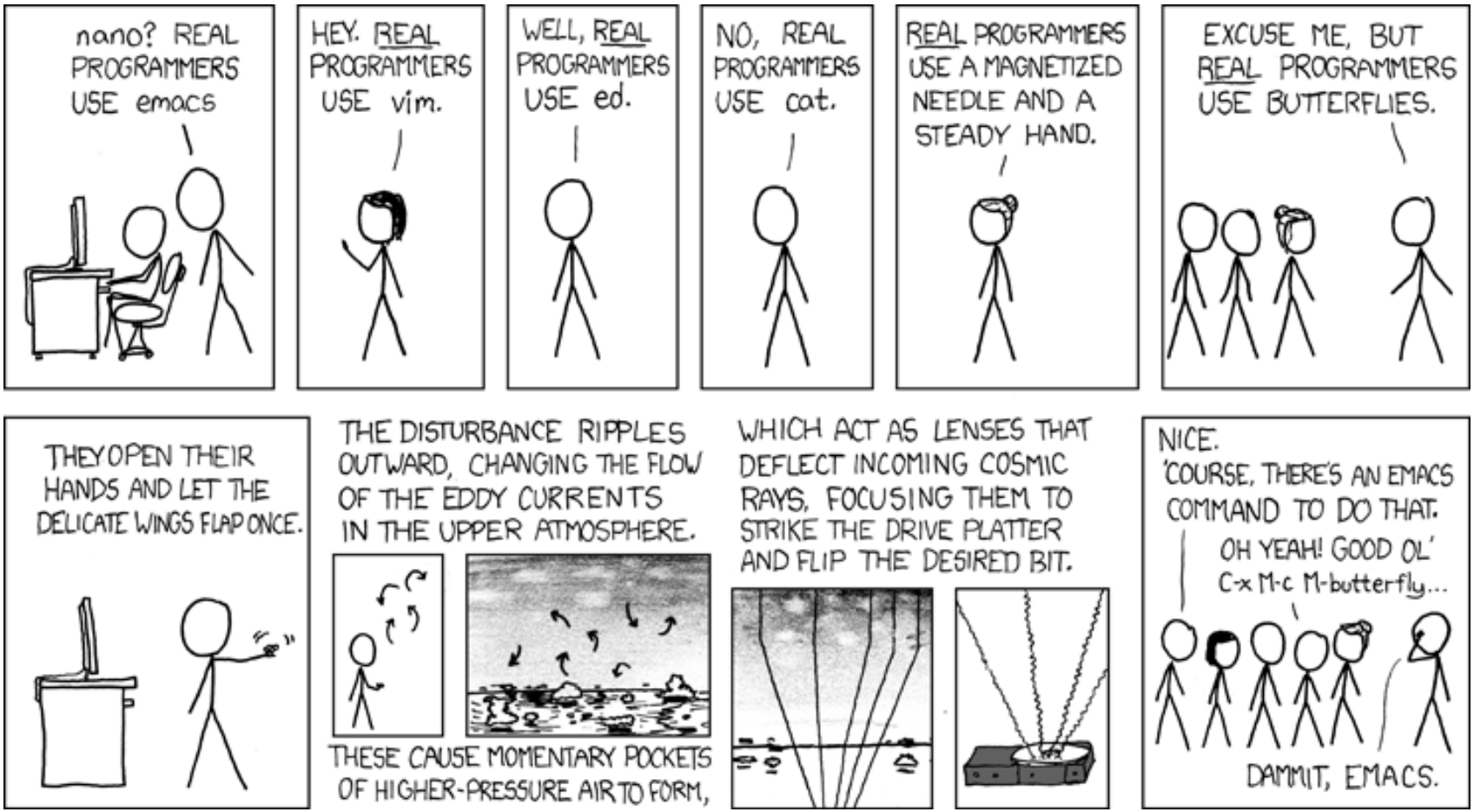
Answers to Exercise 2

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/

Správa verzí

- 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“
  - automatizavaná správa verzí
  - podpora týmové práce
  - mnoho různých řešení: `git`, `cvs`, `svn`, ...

Git

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

Python moduly

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

Python knihovny, frameworky

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

Regulární výrazy

nástroj pro hledání „vzorů“ v textu

- programování
- textové editory
- příkazová řádka: např. `grep`
- teorie: formální jazyky, konečné automaty

Regulární výrazy

- obecně používaný nástroj
- syntax velmi podobná ve většině jazyků, prostředí
- bohatá syntax
- následuje „ochutnávka“, ukázky základního využití v Pythonu

http://xkcd.com/208/

Znaky a speciální znaky

- základní znak „vyhoví“ právě sám sobě
- speciální znaky: `.` `^` `$` `*` `+` `?` `{` `}` `[` `]` `\` `|` `(` `)`
  - umožňují konstrukci složitějších výrazů
  - chceme, aby odpovídaly příslušnému symbolu $\Rightarrow$ prefix \

Skupiny znaků

`[abc]` – jeden ze znaků `a, b, c`
`[^abc]` cokoliv jiného než `a, b, c`

`\d`	Čísla: `[0-9]`
`\D`	Cokoliv kromě čísel: `[^0-9]`
`\s`	Bílé znaky: `[ \t\n\r\f\v]`
`\S`	Cokoliv kromě bílých znaků: `[^ \t\n\r\f\v]`
`\w`	Alfanumerické znaky: `[a-zA-Z0-9_]`
`\W`	Nealfanumerické znaky: `[^a-zA-Z0-9_]`

Speciální symboly

.	libovolný znak
^	začátek řetězce
$	konec řetězce
|	alternativa – výběr jedné ze dvou možností

Opakování

`*`	nula a více opakování
`+`	jedno a více opakování
`?`	nula nebo jeden výskyt
`{m,n}`	m až n opakování

Pozn. `*`, `+` jsou „hladové“, pro co nejmenší počet opakování `*?`, `+?`

Příklady

Jaký je význam následujících výrazů?

- `\d[A-Z]\d \d\d\d\d`
- `\d{3}\s?\d{3}\s?\d{3}`
- `[a-z]+@[a-z]+\.cz`
- `^To:\s*(fi|kit)(-int)?@fi\.muni\.cz`

Regulární výrazy v Pythonu

- knihovna `re` (`import re`)
- `re.match` – hledá shodu na začátku řetězce
- `re.search` – hledá shodu kdekoliv v řetězci
- (`re.compile` – pro větší efektivitu)
- „raw string“ – r'vyraz' – nedochází k interpretaci speciálních znaků jako u běžných řetězců v Pythonu

Regulární výrazy v Pythonu: práce s výsledkem

- match/search vrací „MatchObject“ pomocí kterého můžeme s výsledkem pracovat
- pomocí kulatých závorek () označíme, co nás zajímá

Regulární výrazy v Pythonu: práce s výsledkem

```
>>> m = re.match(r"(\w+) (\w+)", \
                 "Isaac Newton, fyzik")
>>> m.group(0)
'Isaac Newton'
>>> m.group(1)
'Isaac'
>>> m.group(2)
'Newton'
```

Regulární výrazy: xkcd

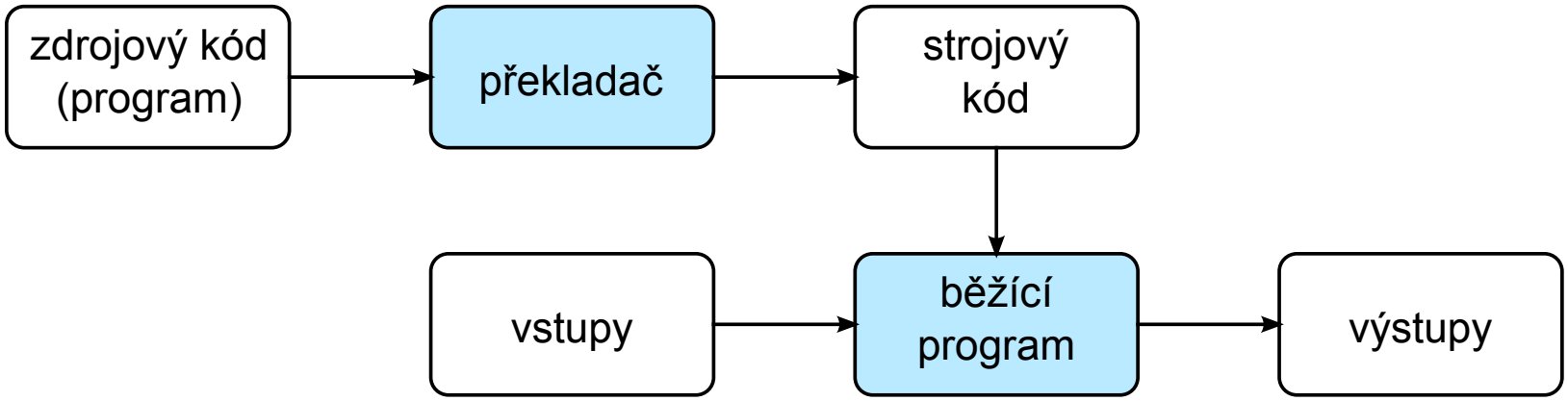
http://xkcd.com/1313/
http://www.explainxkcd.com/wiki/index.php/1313:_Regex_Golf
https://regex.alf.nu/

Programovací jazyky

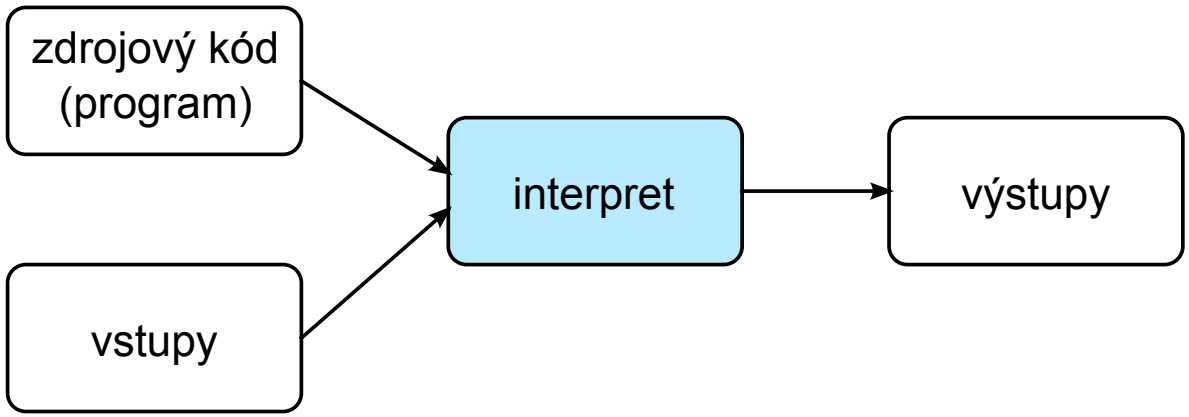
- přehled
- klasifikace, vlastnosti
- historie
- způsoby užití
- neseriózní postřehy

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

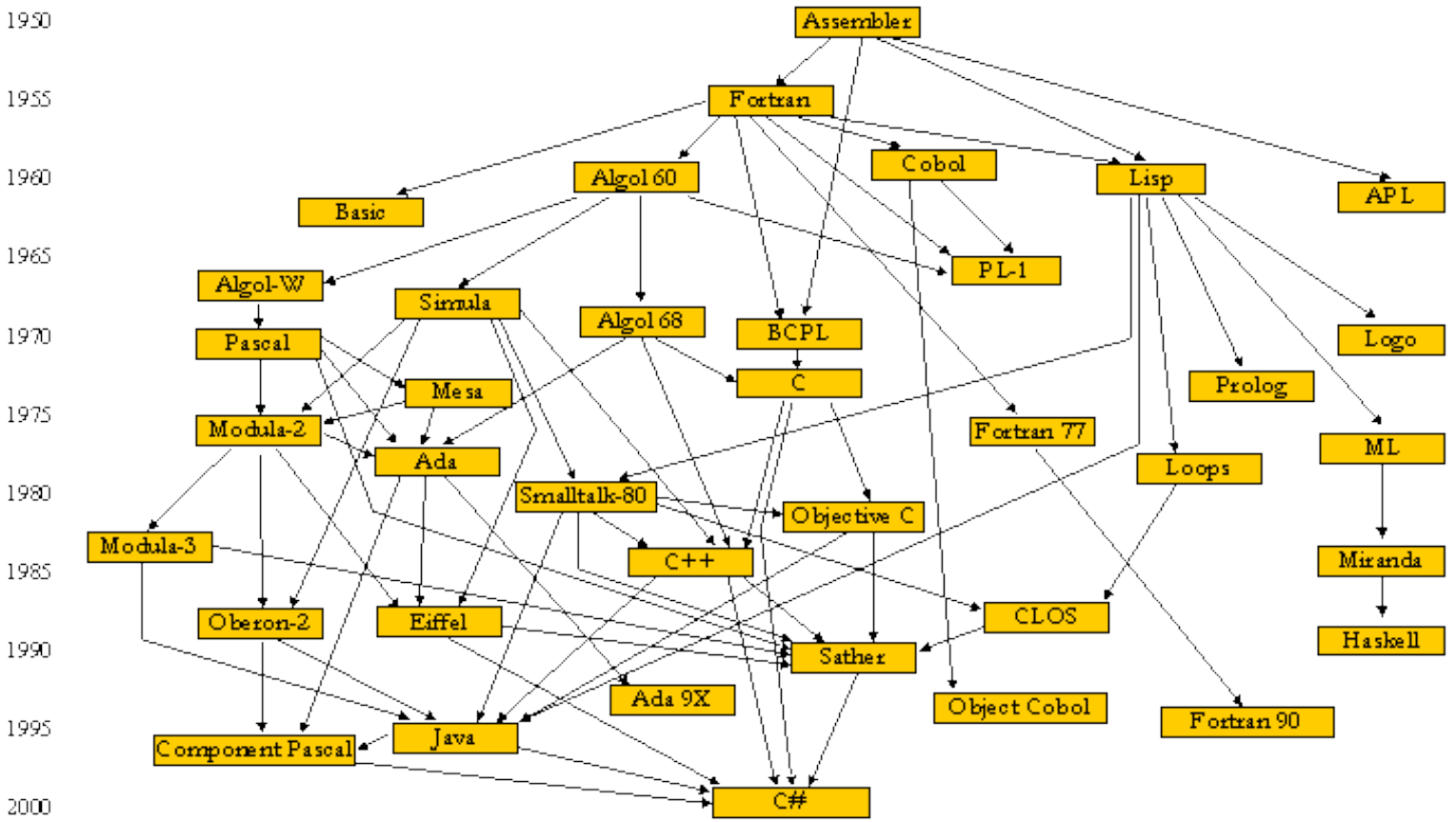
nikoliv dvě kategorie, ale plynulý přechod; zjednodušeno

Programovací jazyky: klasifikace II

zjednodušená klasifikace a použití

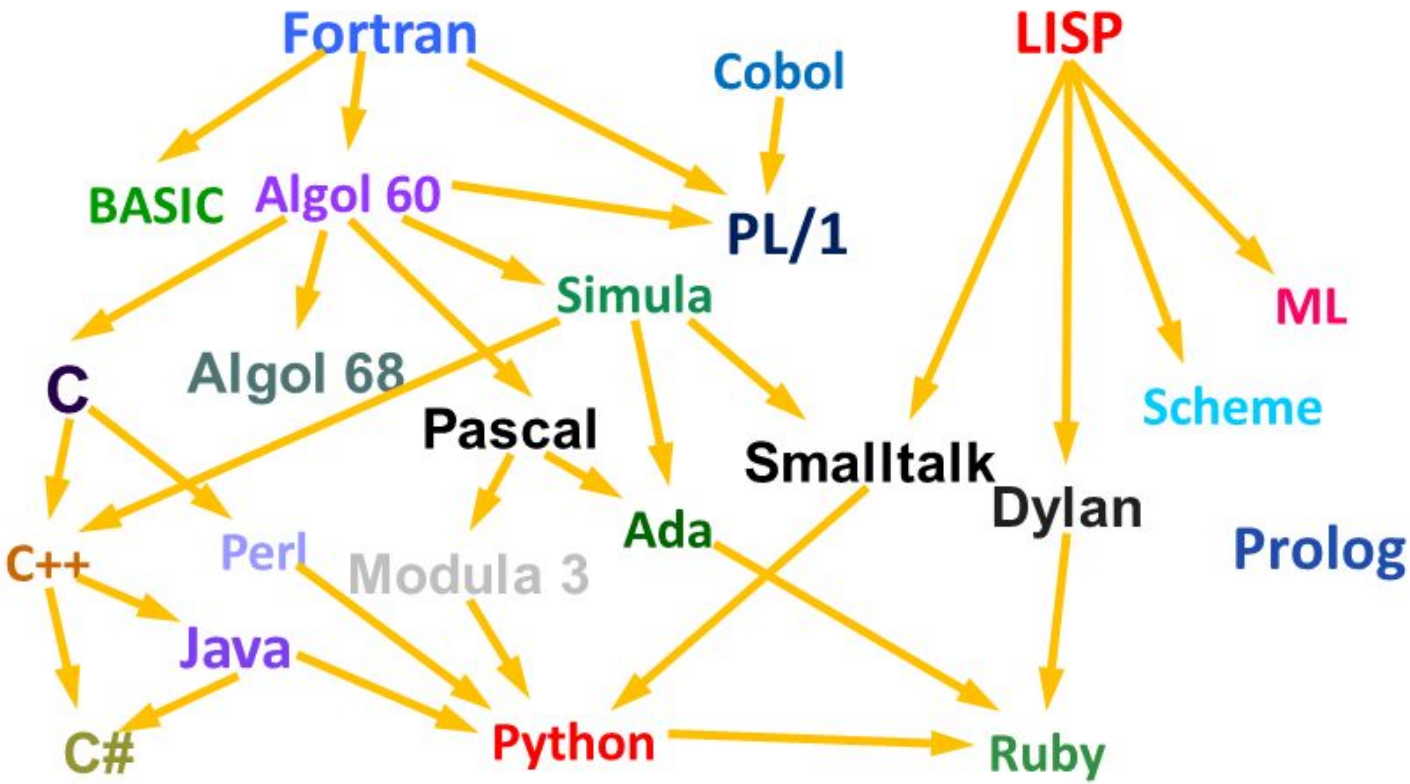
nízko-úrovňové	C, FORTRAN, ...
	vestavěné systémy, rychlé výpočty
objektové	C++, Java, C#, ...
	klasické aplikace, rozsáhlé systémy
skriptovací	Python, PHP, JavaScript, Perl, ...
	programování pro web, skriptování, prototypy
deklarativní	Prolog, LISP, Haskell, ...
	umělá inteligence

Programming Language Family Tree



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, asembler

Historie programovacích jazyků

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

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

BASIC

- „jazyk pro začátečníky“
- rozšířen v 70. a 80. letech na „mikropočítačích“
- *výborný jazyk pro vytvoření špatných programátorských návyků*

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

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.

C: užití

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

C syntax

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

```c
#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++

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

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, ...

Eval

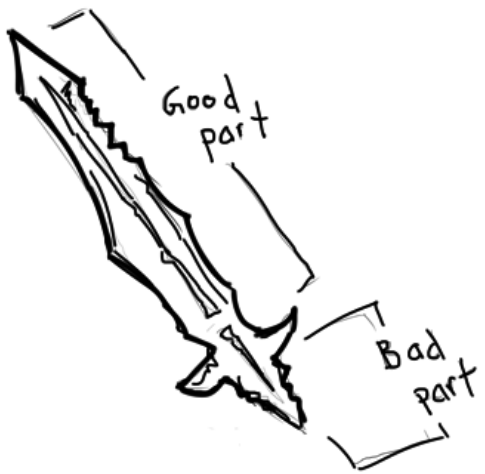
interpretované jazyky flexibilnější než kompilované

typická ukázka: příkaz `eval`

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

JavaScript

JavaScript is a sword without a hilt.



JavaScript

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

PHP

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

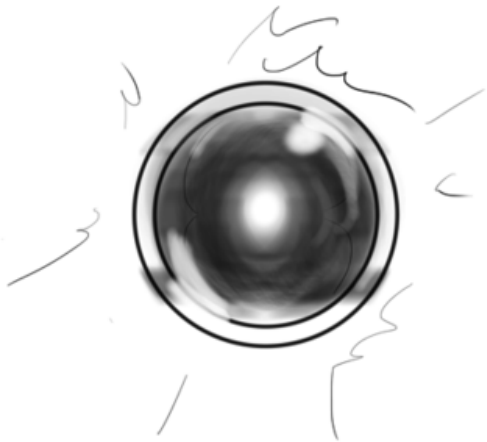
Deklarativní jazyky

- 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

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

Haskell

https://xkcd.com/1312/

Haskell

```
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
```

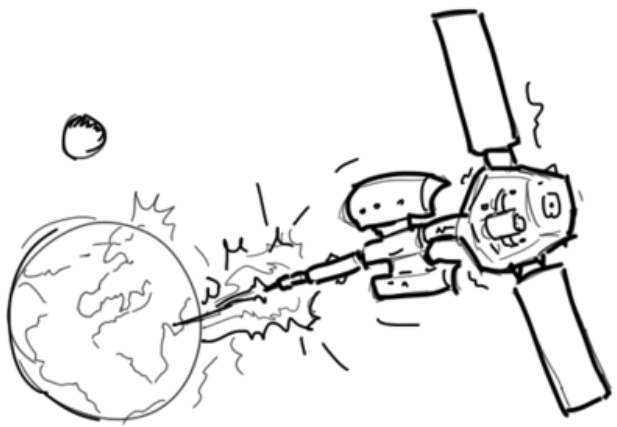
Matematický software

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

- Mathematica
- R

Mathematica

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



Esoterické programovací jazyky

„karikatury“ programovacích jazyků – dobrá karikatura umožňuje uvědomit si důležité aspekty originálu

- Brainfuck
- AsciiDots
- LOLCODE
- OSTRAJava
- Whitespace
- ...

```
++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++.>+.+++++++..+++.>++.
<<++++++++++++++.>.+++.------.--------.>+.>.
```

```
/#$<.
*-[+]
\#1/
```

```
HAI
CAN HAS STDIO?
VISIBLE "HAI WORLD!"
KTHXBYE
```

```
banik pyco
tryda Ostrava{
    rynek(){
        Konzola.pravit("Toz vitaj") pyco
    }
}
fajront pyco
```

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!

Kolik jazyků umíš, tolikrát jsi programátorem.

Popularita jazyků

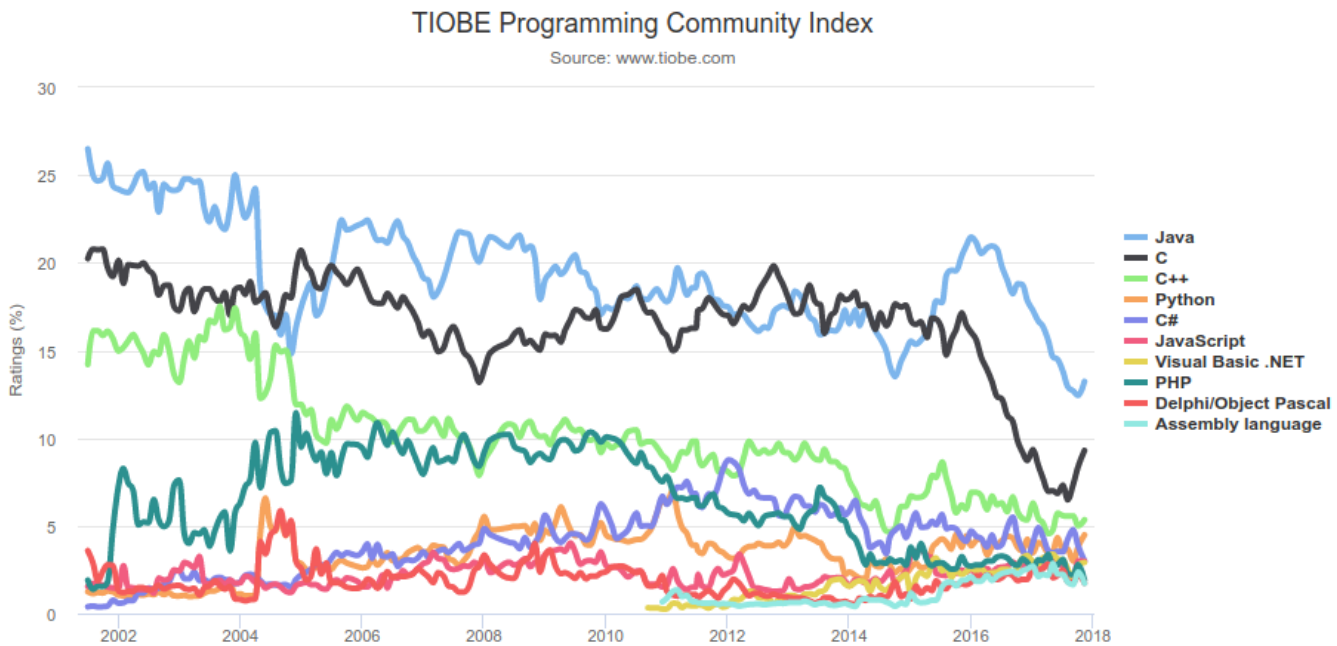
těžko měřitelné, existuje řada pokusů různými metrikami:

- 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 paradigmata
  - **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!