1. Úvod do programování

BAB37ZPR – Základy programování

Stanislav Vítek

Katedra radioelektroniky
Fakulta elektrotechnická
České vysoké učení v Praze

Přehled témat

- Část 1 – O předmětu
  - Organizace předmětu
  - Dostupné prostředky
  - Studijní výsledky
- Část 2 – O programování
  - Než začneme programovat
  - První program

Část I

O předmětu

I. O předmětu

Předmět a lidé

- Webové stránky předmětu
  `https://cw.fel.cvut.cz/b191/courses/bab37zpr`
- Moodle – kopie webu na CW
  `https://moodle.fel.cvut.cz`
- Přednášející a garant předmětu
  - Stanislav Vítek, viteks@fel.cvut.cz
    `http://mmtg.fel.cvut.cz/personal/vitek/`
- Cvičící
  - Martin Řimnáč, rimnacm@fel.cvut.cz

Cíle předmětu

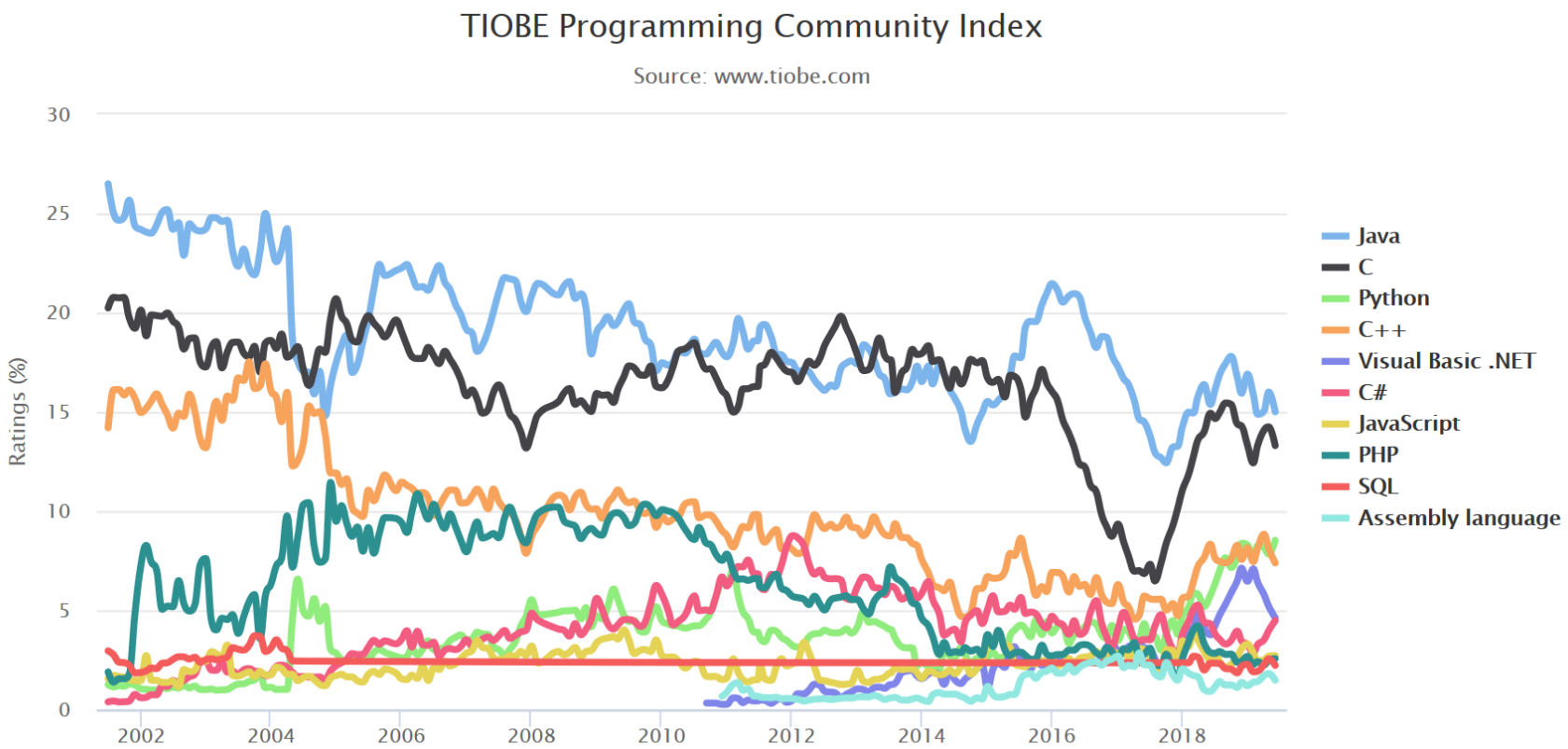
- Motivovat k programování
  - Programování je klíčová dovednost, která může hrát rozhodující roli na trhu práce
- Naučit se algoritmizovat
  - Formulace problému a návrh řešení
  - Rozklad problému na dílčí úlohy
  - Identifikace opakujících se vzorů
- Získat zkušenosti s programováním
  - Základní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

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

    Cvičení, domácí úkoly, zkouška
  - Povědomí o tom, jaké úlohy lze výpočetně řešit

    Programátorovi nestačí perfektní znalost programovacího jazyka, ale především musí vědět, jak vůbec danou úlohu řešit
  - Hledání chyb, práce s dokumentací

Proč Python?



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

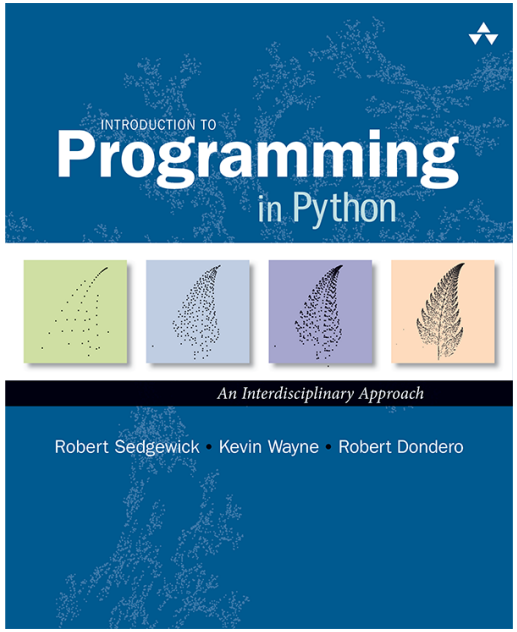
Organizace a hodnocení předmětu

- **BAB37ZPR** – Základy programování
- Rozsah: 2p+2c Zakončení: Z, ZK Kredity: 6

- **Studijní výsledky**
  - Průběžná práce v semestru – domácí úkoly a testy
  - Semestrální práce a zkouškový test
- **Docházka**
  - Přednášky jsou nepovinné, ale velmi doporučené
  - Cvičení jsou povinná, možné dvě omluvené absence

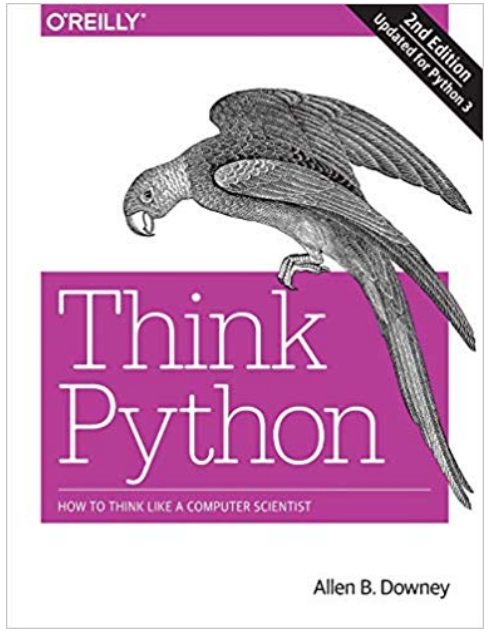
    Na cvičení je třeba se **připravit**, nejlépe návštěvou přednášky a studiem podkladů (příklady)
- **Řešení problémů**
  - Obracejte se na svého cvičícího
  - Při komunikaci e-mailem pište vždy ze své fakultní adresy
  - Do předmětu zprávy uvádějte zkratku předmětu ZPR
  - V případě zásadních problémů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

Zdroje a literatura



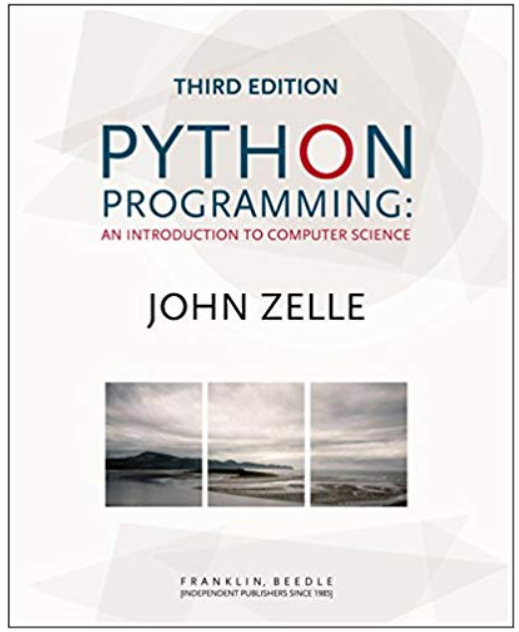
Robert Sedgewick
Introduction to Programming in Python: An Interdisciplinary Approach
Addison-Wesley
2015

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 Introduction to Computer Science
Franklin, Beedle & Associates
2016

ISBN 978-1590282755

I. O předmětu

Služby akademické sítě

- SVTI – `http://svti.fel.cvut.cz/cz`
- Diskové úložiště ownCloud – `https://owncloud.cesnet.cz`
- Zasílání velkých souborů – `https://filesender.cesnet.cz`
- Rozvrh a termíny – `https://portal.fel.cvut.cz`
- FEL Google Account – `http://google-apps.fel.cvut.cz`

  autentizovaný přístup do Google Apps for Education
- Gitlab FEL – `https://gitlab.fel.cvut.cz`
- Přístup k informačním zdrojům – `https://dialog.cvut.cz`

  IEEE Xplore, ACM, Science Direct, Springer Link
- Akademické a kampusové licence – `https://download.cvut.cz`
- MetaCentrum – `http://www.metacentrum.cz`

  Národní Gridová Infrastruktura

I. O předmětu

Domácí úkoly

- Samostatná práce s cílem osvojit si praktické zkušenosti

  Průběžná práce a řešení úkolů.

- Jednotné zadání na prednášce a jednotný termín odevzdání
- Odevzdání domácích úkolů prostřednictvím systému BRUTE

  `https://cw.felk.cvut.cz/upload`

  - Nahrání archivu s nezbytnými zdrojovými soubory
  - Penalizace za pozdní odevzdání
  - Detekce plagiátů

  Cílem řešení úkolů je získat vlastní zkušenost

  - Úkoly jsou jednoduché a navrhované tak, aby byly stihnutelné

Pokud nečemu nerozumíte, ptejte se!

Hodnocení

Zdroj bodů	Maximum	Nutné minimum
Domácí úkoly	30	20
Testy v semestru	20	10
Semestrální práce	20	10
Zkouškový test	30	10
Součet	100	50

- Za práci v semestru je třeba získat nejméně 40 bodů, všechny domácí úkoly musí být odevzdány a to nejpozději do 12.1.2020 ve 23:59 CET!
- Testy v semestru – na počítači, schopnost pochopit problém a napsat krátký program
- Zkouškový test – na papír, otázky teoretického charakteru, jednoduché příklady

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	23.9.
2.	Základní řidící struktury	30.9.
3.	Algoritmy pracující s čísly	7.10.
4.	Řetězce a seznamy	14.10.
5.	Práce s daty, texty	21.10.
6.	Vyhledávání a řazení	4.11.
7.	Proměnné, paměť, typy	11.11.
8.	Datové typy	18.11.
9.	Objekty	25.11.
10.	Knihovny	2.12.
11.	Studentská volba 1	9.12.
12.	Studentská volba 2	16.12.
13.	Zkouškový test	6.1.

Část II

O programování

II. O programování

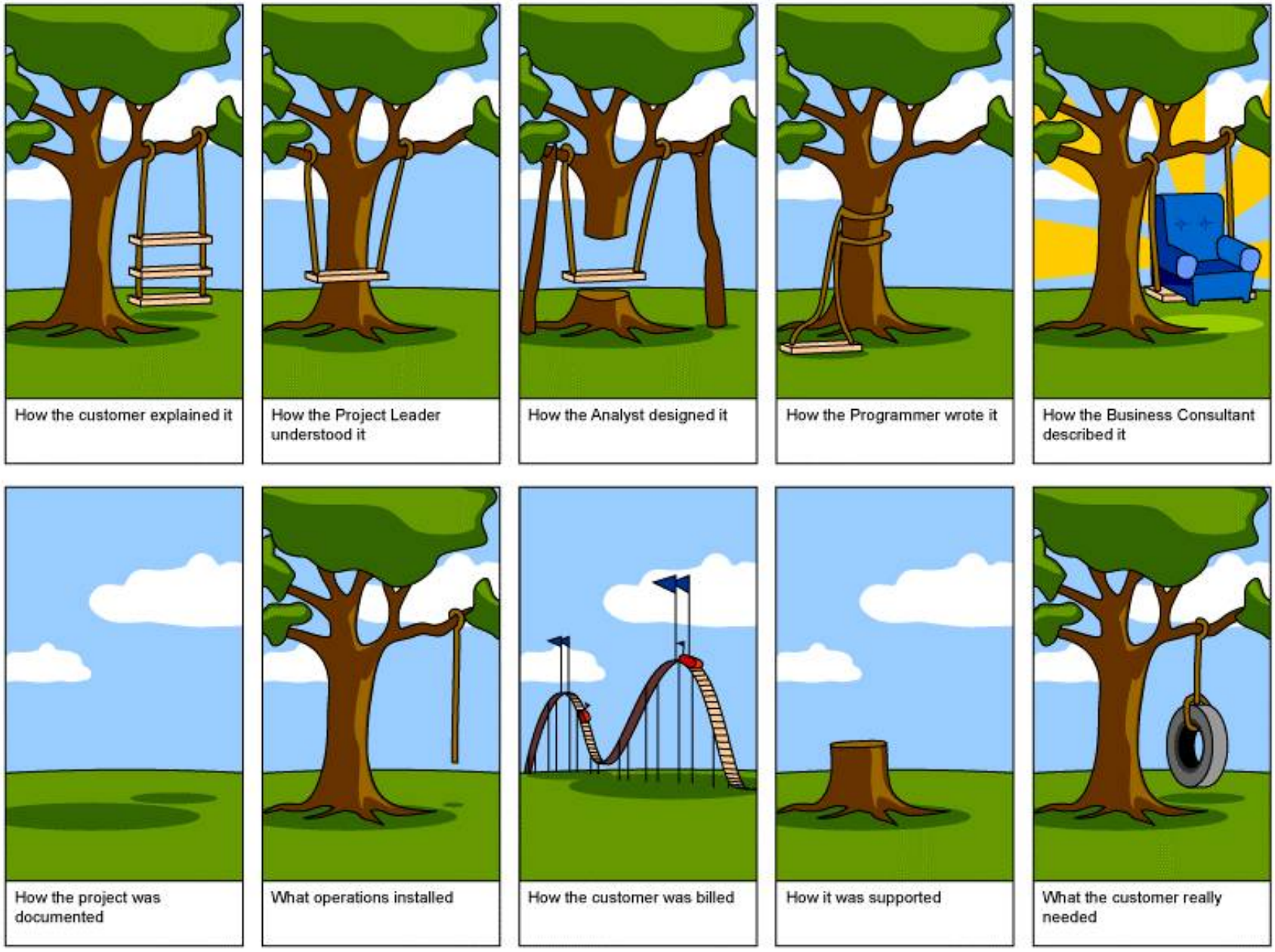
Než začneme programovat

První program

Řešení problémů

1. formulace problému
2. analýza možných řešení a návrh algoritmu
3. implementace v programovacím jazyce
4. testování, ověření funkčnosti
5. optimalizace
6. oprava chyb
7. implementace nových požadavků, údržba
8. dokumentace

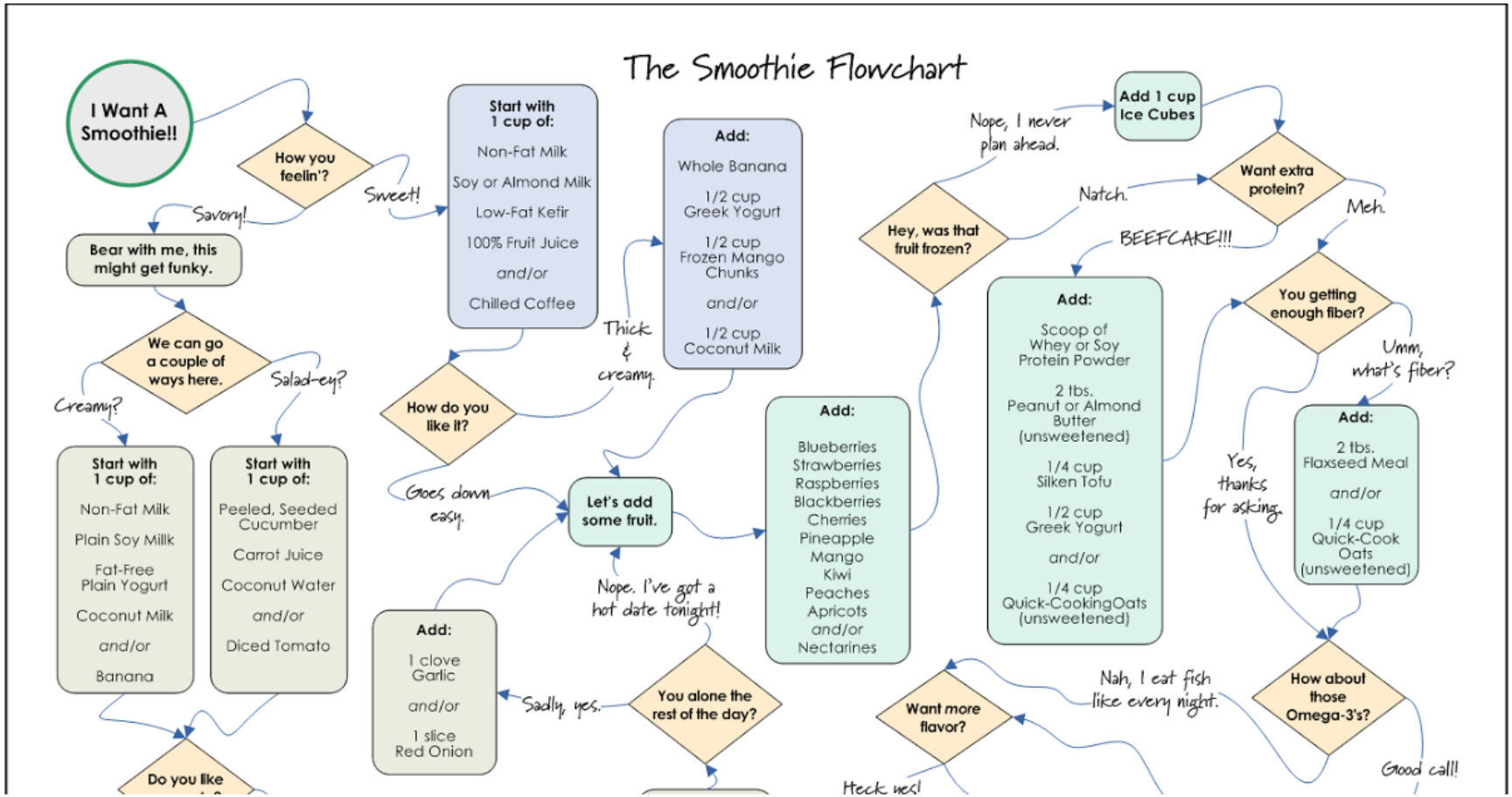
Řešení problémů



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, jak provést určitou činnost.
- Vlastnosti algoritmu:
  1. Skládá se z konečného počtu jednoduchých činností – kroků.

     Je elementární.
  2. Po každém kroku lze určit, jak se má pokračovat nebo skončit.

     Je determinovaný.
  3. Počet opakování jednotlivých kroků algoritmu je vždy konečný.

     Je konečný.
  4. Vede ke správnému výsledku.

     Je rezultativní.
  5. Algoritmus lze použít k řešení celé (velké) skupiny podobných úloh.

     Je hromadný.

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

Základní složky algoritmu

- V případě programování jde zpravidla o transformaci množiny vstupních dat na množinu dat výstupních.
- Kombinace základních složek algoritmu umožňuje vytvářet komplexní programy.

  Posloupnost (sekvence) – tvořena jedním nebo několika kroky, které se provedou právě jednou v daném pořadí.

  Cyklus (iterace) – opakování nějaké posloupnosti, dokud je splněna podmínka opakování.

  Větvení (podmíněná operace) – volba posloupnosti instrukcí na základě vyhodnocení podmínky.

Pokud se některé části algoritmu opakují, je vhodné poslupnosti orgranizovat do větších celků: procedur a funkcí (podprogramů).

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 nebo konstrukčním popisem geometrické úlohy)
  - **programem** – kroky algoritmu jsou popsány instrukcemi procesoru, resp. převedeny z vyššího programovacího jazyka, tedy algoritmus programujeme
- Návrhy algoritmů:
  - **shora dolů** – problém rozdělíme na několik podúloh, které řešíme a spojením dostaneme celý algoritmus
  - **zdola nahoru** – z triviálních úloh skládáme vyšší úlohy a spojením dostaneme celý algoritmus
  - **kombinace obou metod**

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

Programování je když...

- Programování je schopnost samostatně
  - tvořit programy
  - dekomponovat úlohy na menší celky
  - sestavovat z dílčích částí větší programy řešící komplexní úlohu
- Jak začít?
  - Scratch – MIT Media Lab
    `https://scratch.mit.edu/`
  - Angry Birds
    `https://studio.code.org/hoc/1/`
  - Code with Anna and Elsa
    `https://studio.code.org/s/frozen/`

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.
>>>
```

- nebo spustíme IDE prostředí jako `idle`, nebo Jupyter notebook...

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)*

Proměnné a přiřazení

Hodnotu výrazu lze uložit pro pozdější použití

identifikátor = *výraz*

- Příklad:

```
>>> a = 3
>>> b = 3 + a
```

- Jaká je hodnota proměnné b?

```
>>> b
6
```

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

II. O programování

Než začneme programovat

První program

První program

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

```
1 # Vypise pozdraveni
2 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.
```