Základy algoritmizace

1. Úvod

doc. Ing. Jiří Vokřínek, Ph.D.
Katedra počítačů
Fakulta elektrotechnická
České vysoké učení technické v Praze

Základy algoritmizace

- O čem je předmět?
  - Naučit se myslet algoritmicky

Algoritmus řeší problém

  - Naučit se ovládat počítač – programovat v Pythonu (trochu)

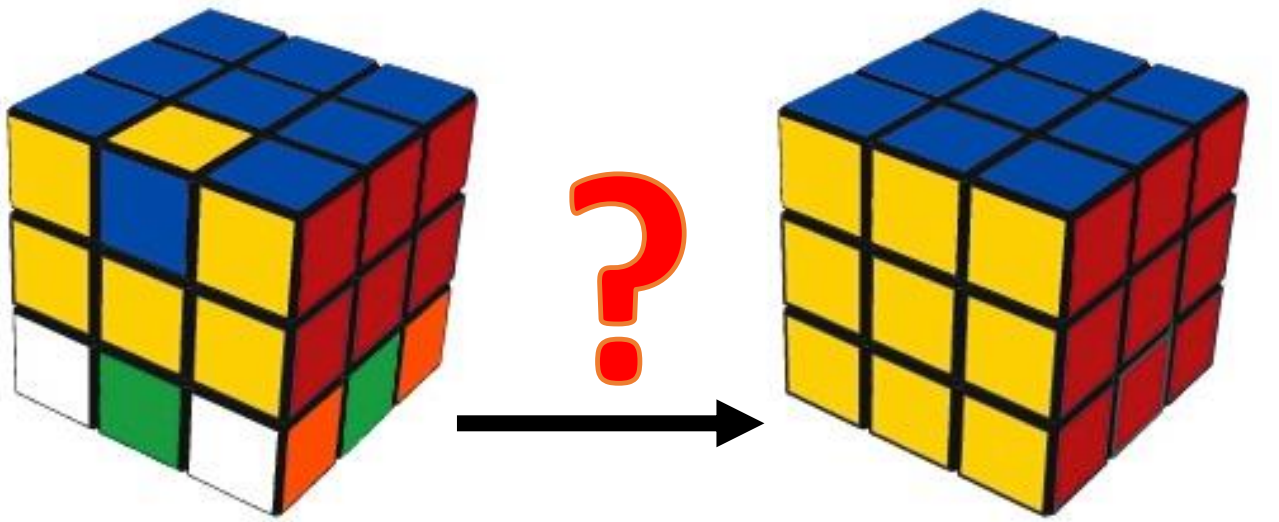
Program řídí počítač

  - Získat přehled o základních algoritmech a práci s daty

Jak funguje moje navigace?

Source: http://www.marcinkossakowski.com/wp-content/uploads/2014/11/dijkstra-map.jpg

Základy algoritmizace



Algoritmus!

→ program

Source: http://www.qedcat.com/rubiks_cube/

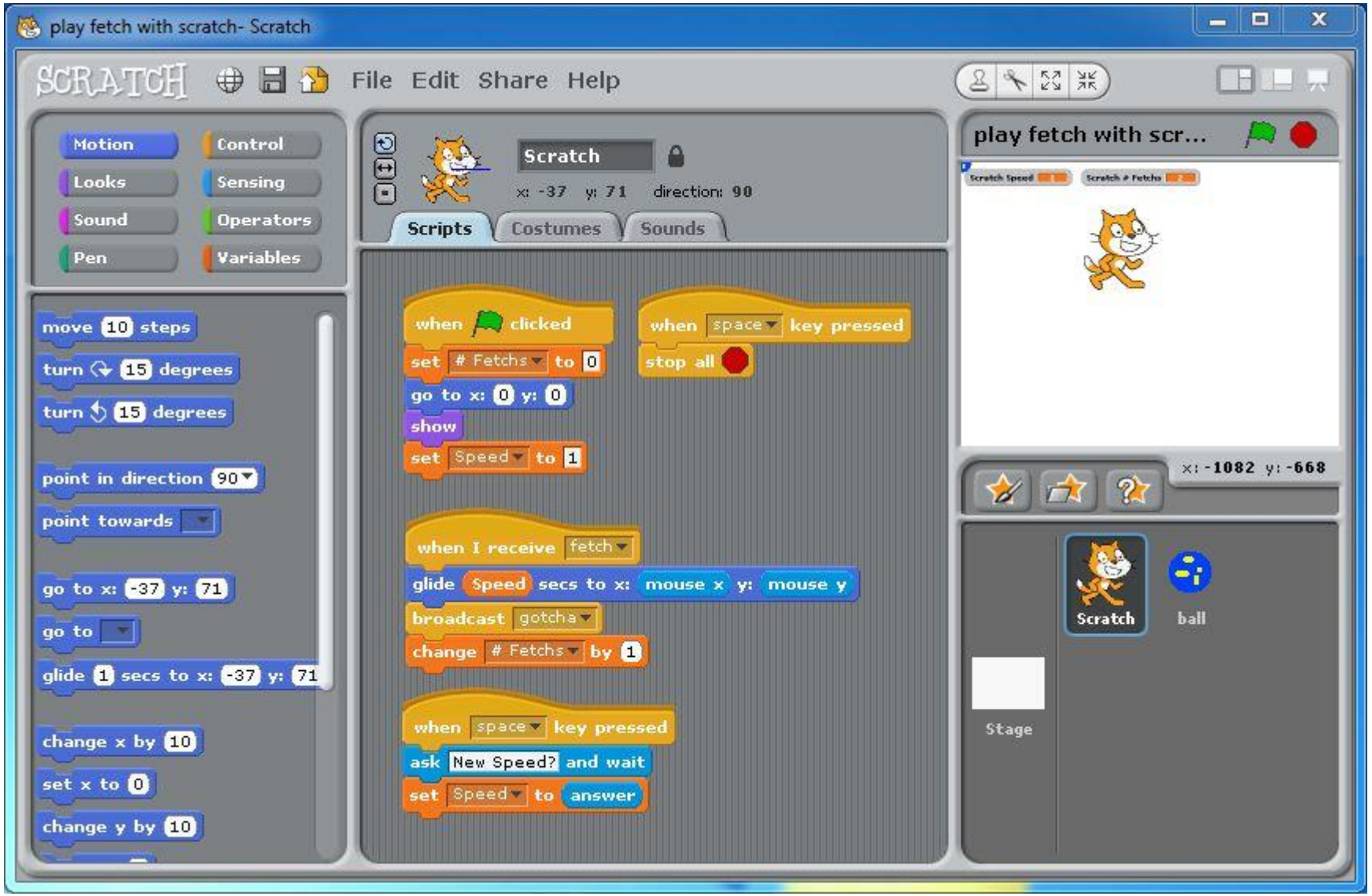
Základy algoritmizace

Computer **programming** (often shortened to **programming**) is a process that leads from an original formulation of a computing problem to executable computer programs.

Computer programming - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/Computer_**programming** Wikipedia

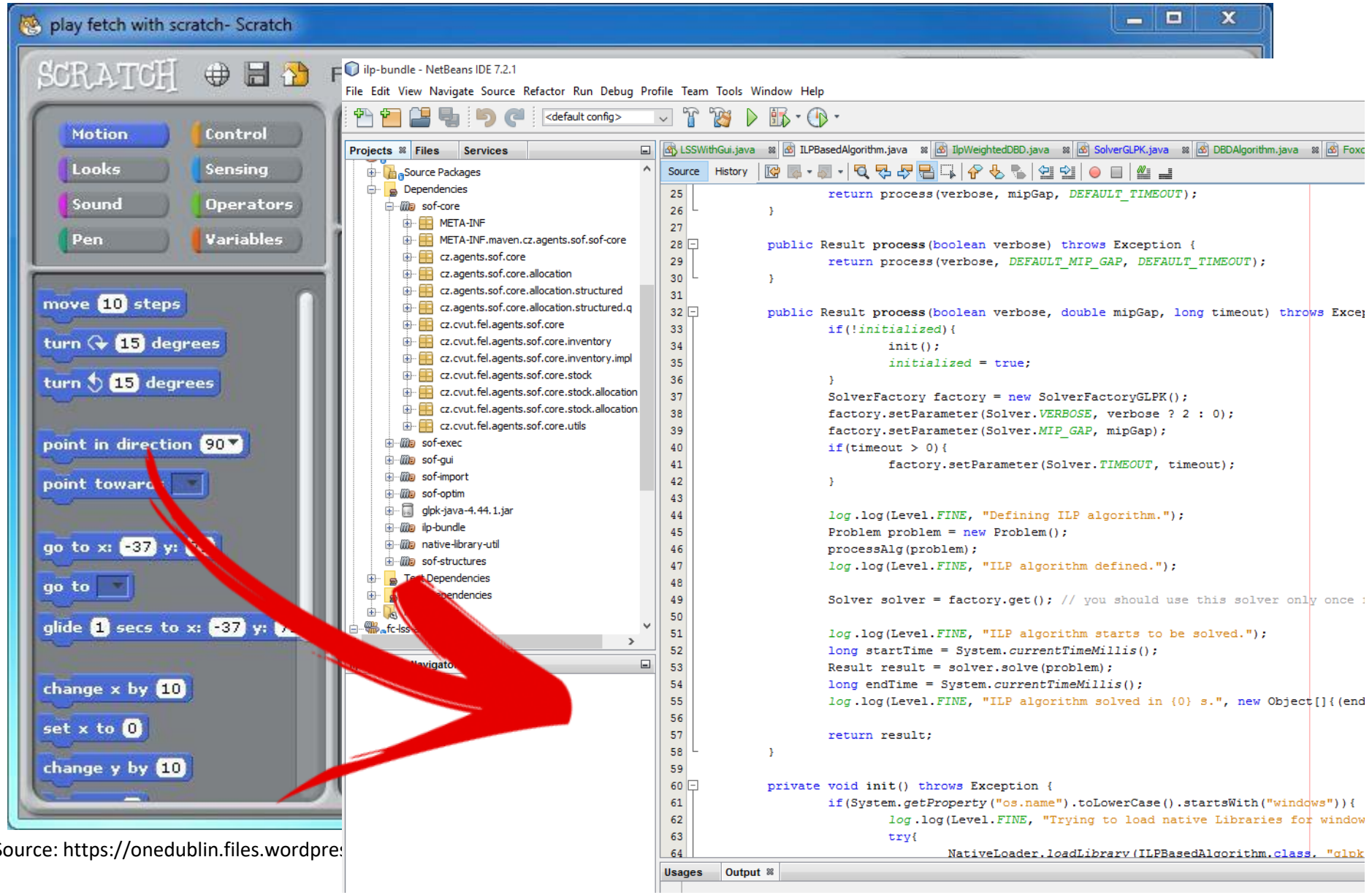
More about Programmer

Základy algoritmizace



Source: https://onedublin.files.wordpress.com/2010/01/scratch-computer-programming-example.jpg

Základy algoritmizace



Source: https://onedublin.files.wordpres

Základy algoritmizace

- Zkuste si: code.org

Základy algoritmizace

- **Dnes**
  - Organizace předmětu
  - Programování a výpočty
  - Python

Organizace předmětu

Organizace předmětu

- Stránky předmětu

  https://cw.fel.cvut.cz/wiki/courses/b6b36zal/start

- Tutoriály

  http://www.tutorialspoint.com/python

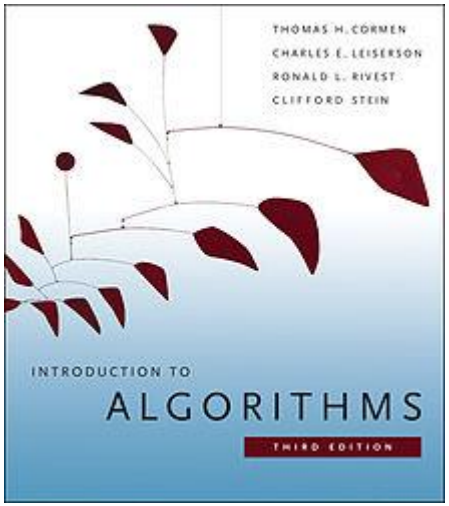
  http://www.py.cz

- Literatura

  Cormen, Leiserson, Rivest and Stein:

  Introduction to Algorithms.

  MIT Press and McGraw-Hill.



Organizace předmětu

- Přednáška
  - Nepovinná, ale doporučená
  - Předpokládáme, že student ZNÁ látku z přednášky
- Cvičení
  - Povinné (detaily na cvičení)
  - 10 úloh za semestr
  - Odevzdávání domácích úkolů

https://cw.felk.cvut.cz/brute/

- Hodnocení
  - 40 bodů úlohy ze cvičení (min. 10 bodů)
  - 60 bodů zkouška
  - A-F za 100 až 50 bodů

Organizace předmětu

- Samostatná práce s cílem osvojit si praktické zkušenosti
- Odevzdání úloh = nahrání nezbytných souborů + ověření správnosti automatickými testy

Detekce plagiátů!

- Úkoly jsou navrženy tak, aby se byly stihnutelné
- Průběžná práce je nejefektivnější

- Pokud něčemu nerozumíte, ptejte se!

- Pokud vám přijde úkolů málo, ptejte se po dalších úlohách na procvičování

Bodové hodnocení úloh

Úloha	Zaměření	Max počet bodů
Introduction	Upload system introduction	0
Python in action	Python introduction	1
Calculator	Number and operations and inteligent	2
PI number	Calculation PI number using cycles	3
Polynomials	Using array to calculate and evaluate polynoms	4
Data sorting	Sorting array and finding most/less important element	5
Showroom	Linked list in car showroom	7
BST	Binary search tree	5
Permutations	Permutation and recursion	5
Quickest path	Dijkstra	8

Počítačové laboratoře

- Operační systém Linux
- Síťové domovské adresáře

- Přenos a synchronizace souborů
  - USB media
  - Síťové přenosy (ftp, ssh, atp.)
  - Owncloud - https://owncloud.cesnet.cz/

- Vlastní počítače
  - V učebnách je k dispozici Eduroam

Programování a výpočty

Základní koncept programování

„Separating Programming Sheep from Non-Programming Goats”

http://blog.codinghorror.com/separating-programming-sheep-from-non-programming-goats/

- Efektivní metody výuky programování se hledají již od dob prvních počítačů, tj. přes více než 50 let
- Presto se zdá, že je každý základní kurz programování obtížný a 30% až 60% studentů jej na poprvé nezvládne
- Základní koncept je pochopení principu přirazení hodnoty proměnné

Další koncepty jsou rekurze/iterace a „concurrency“

Test pochopení principu přiřazení

- Zápis programu pro přirazení hodnot do proměnných *a* a *b* a následné přirazení proměnné *b* do *a*.

Přirazení hodnoty proměnné

```
1  a = 10
2  b = 20
3  a = b
```

- Jaké jsou hodnoty proměnných *a* a *b*?

a. a = 20 b = 0

b. a = 20 b = 20

c. a = 0 b = 10

d. a = 10 b = 10

e. a = 30 b = 20

f. a = 30 b = 0

g. a = 10 b = 30

h. a = 0 b = 30

i. a = 10 b = 20

j. a = 20 b = 10

Skupiny počítačových uživatelů

„Uživatel“

- Spouštěč programů
- Zadává vstup

Píše, kliká

- Čeká na výstup
- Čte výstup

- Relativně omezená množina vstupů

Pouze to co je dovoleno

„Programátor“

- Spouští programy
- Dává počítači příkazy

Řadí je do posloupnosti

- Kombinuje příkazy
- Vytváří nové programy

- Rozmanitější možnosti použití

Omezen pouze limity počítače

Způsob reprezentace znalostí

- Z hlediska výpočtu můžeme rozlišit dva základní typy znalostí:

Způsob popisu problému

Deklarativní

- Tvrzení popisuje stav
- Axiomatické
- Umožňuje jednoduše ověřovat (testovat) pravdivost tvrzení
- Neposkytuje návod jak vyčíslit hodnotu

Příklad:

$$\sqrt{x} = y, y^2 = x, x \geq 0, y \geq 0$$

Imperativní

- Popisuje jak něco vypočítat
- Posloupnost výpočtu
- Test jak ovlivnit průběh výpočtu

Příklad:

1. If $y^2 \approx x$
2. Then
   return y
3. Else
   $y \leftarrow \frac{y+\frac{x}{y}}{2}$
4. Go to Step 1

Výpočetní prostředky (počítače)

- Jednoúčelové přístroje s předepsaným chováním
  - program / posloupnost kroků (instrukcí) je vestavěná a neměnná

Kalkulačka, pračka, první telefony

- Počítač s uloženým programem v paměti
  - Posloupnost instrukcí čtena z paměti
  - Flexibilita ve tvorbě posloupnosti

Program lze libovolně měnit

  - Architektura počítače se společnou pamětí pro data a program
    - Von Neumannova architektura počítače

John Louis von Neumann (1903–1957)

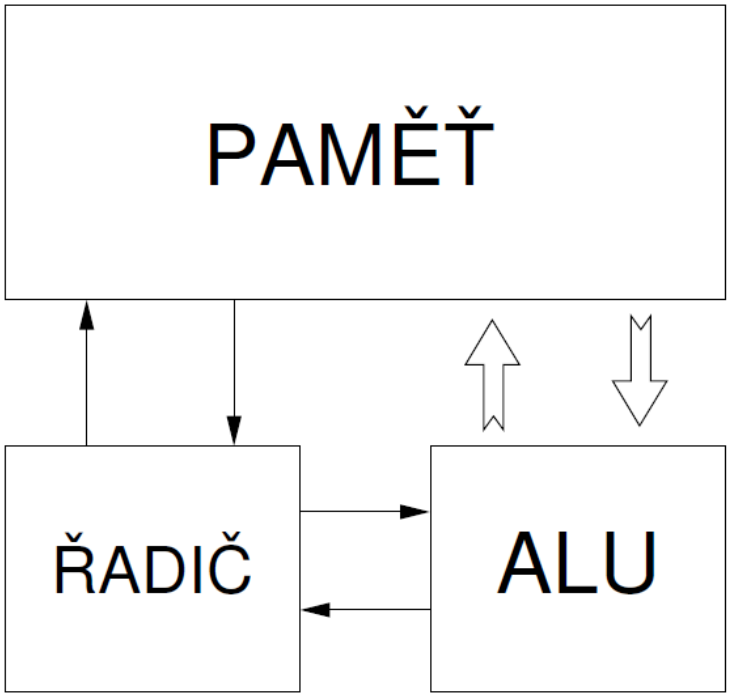
Von Neumannova architektura

- ALU – Aritmeticko-logická jednotka (Arithmetic Logic Unit)

  Základní matematické a logické instrukce

- PC – Čítač instrukcí (Program Counter)

  „Ukazuje" na místo v paměti s instrukcemi pro vykonání



V drtivé většině případů je program posloupnost instrukcí zpracovávající jednu nebo dvě hodnoty (uložené v nějakém paměťovém místě) jako vstup a generování nějaké výstupní hodnoty, kterou ukládá někam do paměti nebo modifikuje hodnotu PC (podmíněné řízení běhu programu).

Assembler

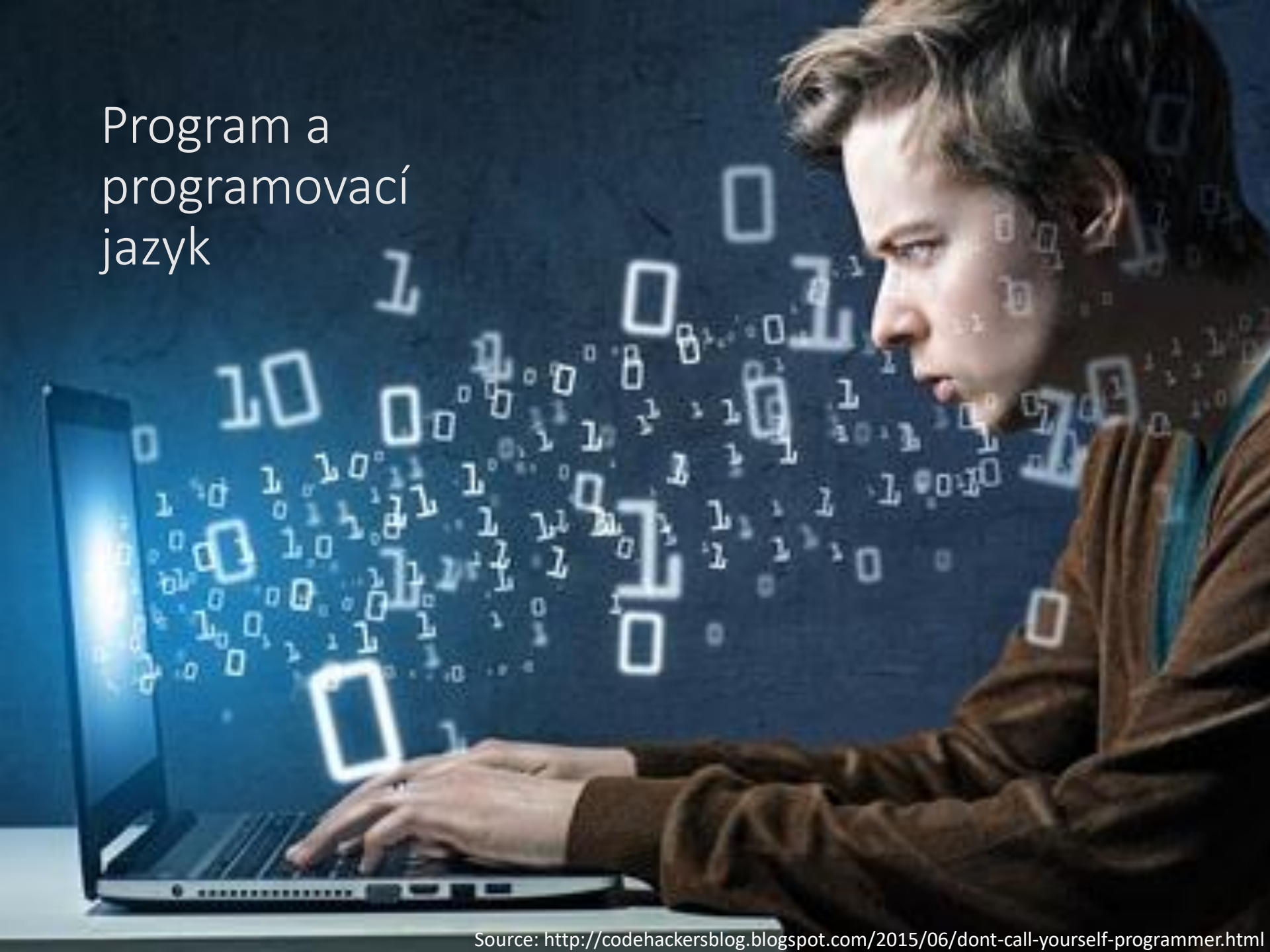
- Jazyk symbolických adres
- Nízko-úrovňový (nad strojovým kódem)

```
; Accepts a number in register AX;
; subtracts 32 if it is in the range 97-122;
; otherwise leaves it unchanged.

SUB32   PROC            ; procedure begins here
        CMP   AX,97     ; compare AX to 97
        JL    DONE      ; if less, jump to DONE
        CMP   AX,122    ; compare AX to 122
        JG    DONE      ; if greater, jump to DONE
        SUB   AX,32     ; subtract 32 from AX
DONE:   RET             ; return to main program
SUB32   ENDP            ; procedure ends here
```

Source: http://www.cybercomputing.co.uk/Languages/Languages/Low_level/assembly.html

Program a programovací jazyk



Source: http://codehackersblog.blogspot.com/2015/06/dont-call-yourself-programmer.html

Algoritmus

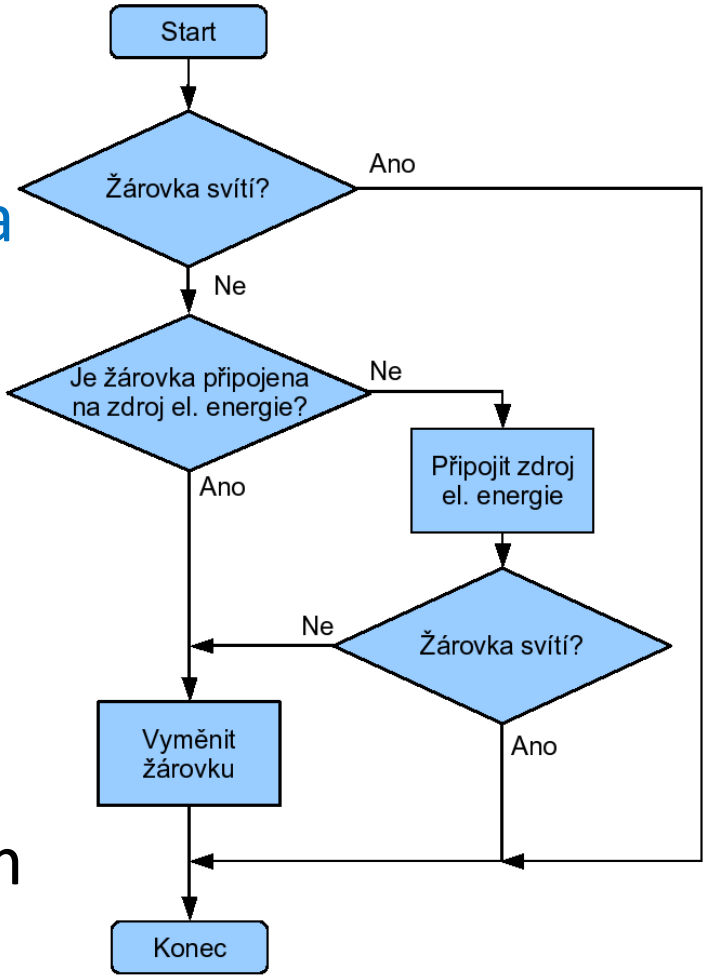
- Vstup → algoritmus → výstup
- Soubor → program → soubor
- Klávesnice → program → obrazovka
- GUI → program → GUI

algoritmus

algoritmus

algoritmus

- Příklad algoritmu - vývojový diagram



Source: https://cs.wikipedia.org/wiki/V%C3%BDvojov%C3%BD_diagram

Program je „recept“

- Program je posloupnost kroků (výpočtů) popisující průběh výpočtu pro řešení problému

(je to „recept” na řešení problému)

- Pro zápis receptu potřebujeme jazyk

Způsob zápisu programu

- Jazyk definuje základní sadu primitiv (operací/příkazu), které můžeme použít pro zápis receptu
- S konečnou množinou primitiv dobrý programátor naprogramuje „cokoliv”.

Co muže být vyjádřeno

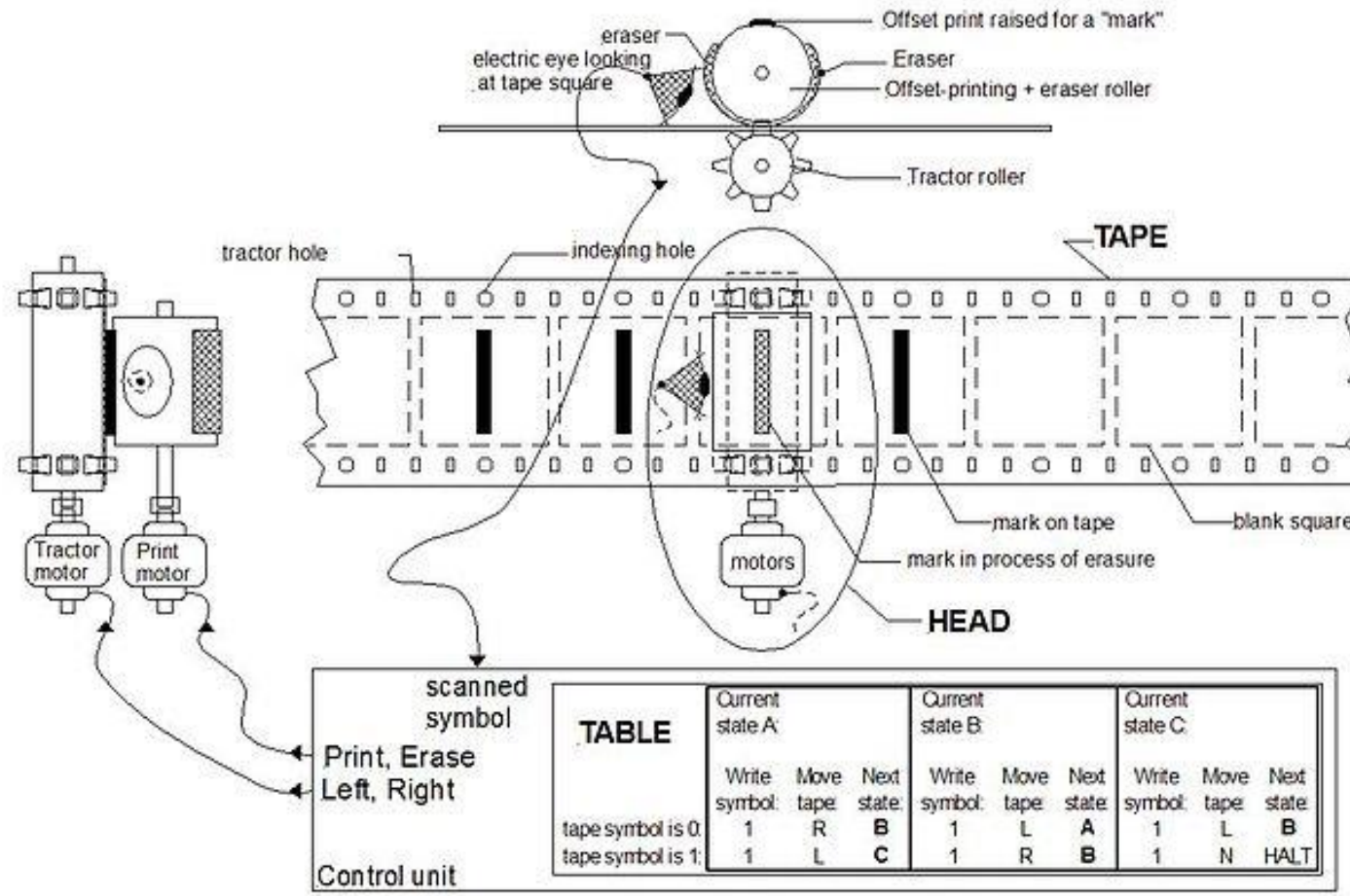
- Turing Machine – obecný model počítacího stroje

Alan Turing, 1936

- V předmětu B6B36ZAL používáme programovací jazyk Python

Programování není o znalosti konkrétního programovacího jazyka, je to o způsobu uvažování a řešení problému

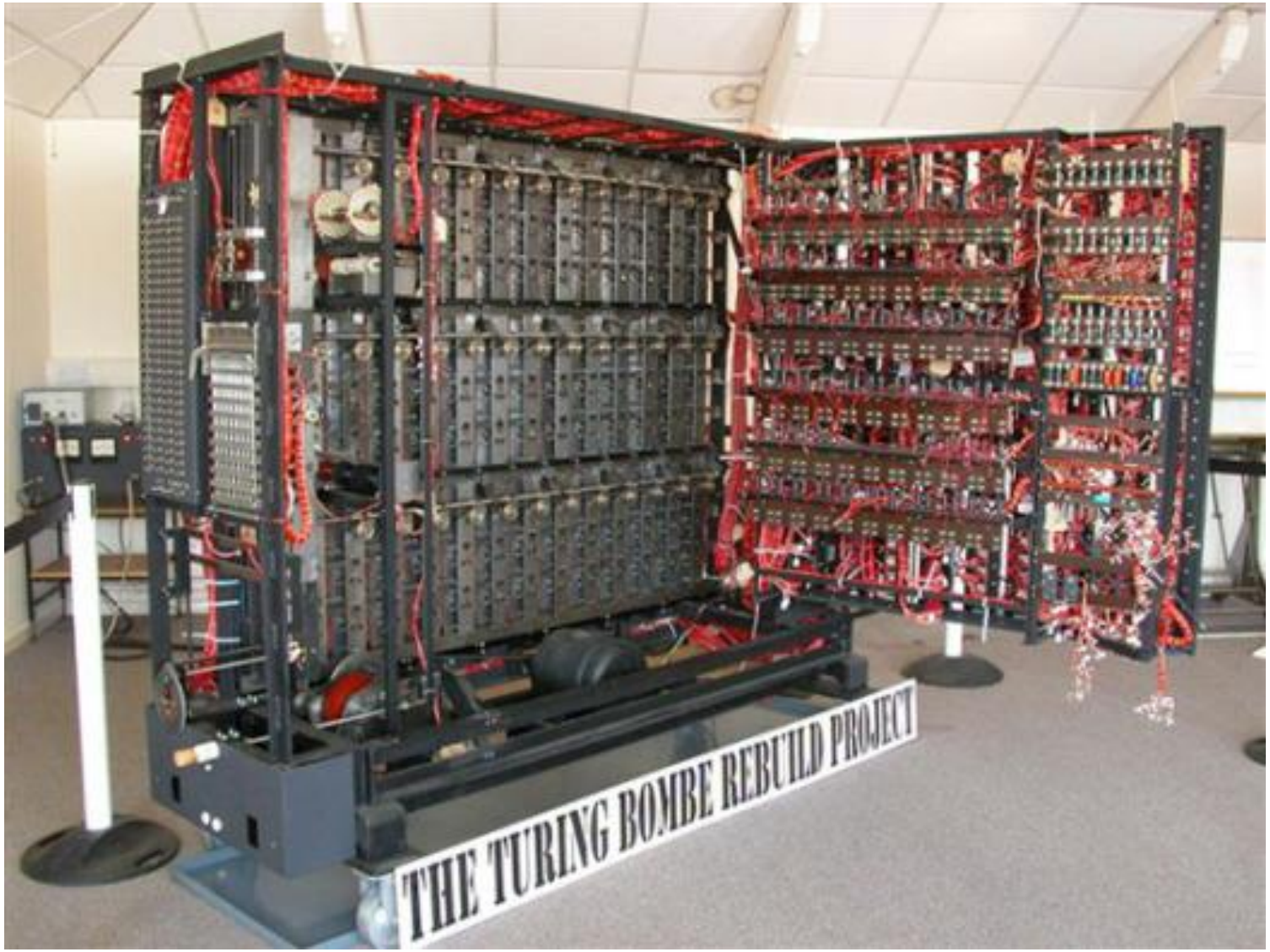
Turing machine



A fanciful mechanical Turing machine's TAPE and HEAD. The TABLE instructions might be on another "read only" tape, or perhaps on punch-cards. Usually a "finite state machine" is the model for the TABLE.

Source: https://en.wikipedia.org/wiki/File:Turing_machine_1.JPG

Turing machine



Source: https://onetimecode.wordpress.com/2015/03/22/the-imitation-game-alan-turing-defeat-enigma-machine-in-world-war-ii/

Programovací jazyk

- Existuje množství programovacích jazyků
- Nelze říci, že jeden jazyk je lepší než druhý

V podstatě jsou všechny ekvivalentní

- Můžeme, ale říci, že některé jazyky se hodí na konkrétní úlohy
- Základní dělení:
  - Vysoko-úrovňové a nízko-úrovňové

Liší se mohutností množiny primitiv

  - Obecné a speciální (určené pro konkrétní aplikace)
  - Interpretované a překládané
  - Dle typu: imperativní (procedurální), funkcionální, logické (deklarativní), objektově-orientované

Definice programovacího jazyka

- Syntax – definice povolených výrazů a konstrukcí programu

  Plná kontrola a podpora vývojových prostředí

  - Príklad popisu výrazu gramatikou v Backus-Naurove formě.

    <exp> ::= <exp> + <exp> | <exp> * <exp> | <exp> |

    <number> <number> ::= <number> <digit> | <digit>

    <digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

- Statická sémantika – definuje jak jsou konstrukty používány

  Částečná kontrola a podpora prostředí

  - Príklad axiomatické specifikace: *{P} S {Q}*, *P*- precondition, *Q*- postcondition, *S* - konstrukce jazyka.

- Plná sémantika – co program znamená a dělá, jeho smysluplnost

  Kontrola a ověření správnosti je kompletně v režii programátora

Správnost programu

- Syntakticky i staticky sémanticky správný program neznamená, že dělá to co od něj požadujeme
- Správnost a smysluplnost programu je dána očekávaným chováním při řešení požadovaného problému
- V zásadě při spuštění programu mohou nastat tyto události:
  - Program havaruje a dojde k chybovému výpisu

    Mrzuté, ale výpis (report) je dobrý start řešení chyby (bug)
  - Program běží, ale nezastaví se a počítá v nekonečné smyčce.

    Zpravidla velmi obtížné detekovat a program ukončujeme po nějaké době
  - Program včas dává odpověď

    Je však dobré vědět, že odpověď je korektní

Správnost programu je plně v režii programátora, proto je důležité pro snadnější ověření správnosti, ladění a hledání chyby používat **dobrý programovací styl**

Program a jeho zápis

- Program – popis činnosti prováděné počítačem
- Programovací jazyk – notační systém pro zápis programu

Program zpravidla zapisujeme ve zdrojových (textových) souborech

- Čitelnost programu:
  - strojová – efektivnost kódu
  - lidská – srozumitelnost, udržovatelnost, kódovací konvence,

Vývojová prostředí (editor), debugger, nástroje pro správu verzí, analýza, testování → ***softwarové inženýrství***

- Abstrakce
  - datová – základní typy, struktury, modulární
  - řídící – základní, strukturální, modulární

Source: http://geoawesomeness.com/wp-content/uploads/2014/08/progLanguages.jpg

Python

Python

- Vysokoúrovňový skriptovací programovací jazyk
- Dynamická kontrola datových typů
- Různá programovací paradigmata
- Open-source, různé platformy
- Jednoduchý na učení
  - V předmětu ZAL Python 3.6

- Online: https://repl.it/languages/python3
- Python: https://www.python.org/downloads/
- **PyCharm EDU:** https://www.jetbrains.com/pycharm-edu
- PyCharm IDE: https://www.jetbrains.com/pycharm/

Licence na download.cvut.cz

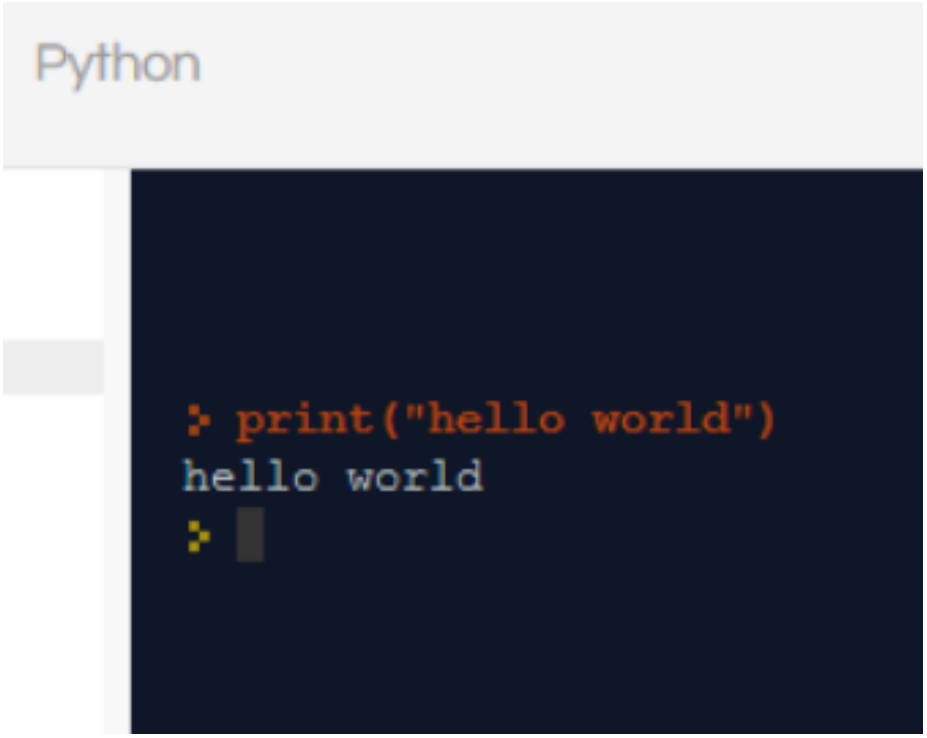
Jak vypadá program?

Source: http://learntocodewith.me/wp-content/uploads/2014/04/python-language.jpg?230531

Příklad

- Hello world – nejčastější „první program“

  > print(“hello world”)



Příklady na https://cw.fel.cvut.cz/wiki/courses/pri-bootcamp/01

Spouštění programu

- Z konzole
  - Pravá část repl.it
  - V příkazové řádce „python“
  - V PyCharm Tools→Python Console
- Ze souboru
  - Levá část repl.it
  - V příkazové řádce „python program.py“
  - V PyCharm „zeleným tlačítkem“

Můžeme programovat!

Výstup

```
print("Hello World!")
print(100)
print(3+6)
print(3*6)
print(6*2-2*4)
print(2*(3+6))
```

```
Hello World!
100
9
18
4
18
```

```
print("3*6")
print(3*6)
```

```
3*6
18
```

```
print("abc"+"def")
print("abc"+123)
```

```
abcdef
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object to str implicitly
```

Komenář

```
# toto je komentar

print("Hello World!") # toto je komentar

"# toto neni komentar"
```

Vstup

- Textový řetězec

```
input("zadej text")
```

Pro zpracování lze uložit do proměnné,

případně přetypovat – viz. příští přednáška

```
vstup = input("Jak se mas?")
print("Ja se mam taky "+vstup)
```

Zkuste si

- Hra s geniální umělou inteligencí

```
print("Je to "+input("Zadej cislo, ktere mam hadat "))
```

Příště proměnné, datové typy a funkce

Source: https://scifi.stackexchange.com/questions/19782/how-does-seeing-the-code-help-neo