

Počítače a programy
(1. přednáška)



FF UK

Department of Logic

Jonathan L. Verner

computer, n.: a programmable usually electronic device that can store, retrieve, and process data (Merriam-Webster)

- ▶ zpracovává data na základě předpisu, který není předem dán! (Babbage)
- ▶ není (princiální) rozdíl, mezi daty a předpisy (von Neumann)!

Nůž na papír je tedy současně předmětem, jenž je vyráběn určitým způsobem a který je na druhé straně jednoznačně užitečný a nelze předpokládat, že by někdo vyráběl nůž na papír, aniž by si byl vědom toho, k čemu bude tento předmět sloužit. Řekneme tedy, že podstata nože - to jest souhrn návodů a kvalit, jež ho umožňují vyrobit a určit k čemu je - ... předchází existenci.

J. P. Sartre: Existencialismus je humanismus

- ▶ Věci: podstata (esence) předchází existenci.
- ▶ Lidé: existence předchází esenci.
- ▶ Počítače: existence předchází esenci ???

programming, v.: to predetermine the thinking, behavior, or operations of a computer (Merriam-Webster)

Je třeba:

- ▶ Naučit se instrukce, kterým počítač rozumí (jazyk počítače).
- ▶ Umět přesně formulovat a analyzovat nejasně zadané problémy.
- ▶ Umět vidět problém z nadhledu a zároveň nezapomenout na detaily.
- ▶ Umět komunikovat s lidmi!
- ▶ Částečně je to umění.
- ▶ Každý (skoro) se to může alespoň trochu naučit.

- ▶ Počítače hrají čím dál tím větší roli, je dobré vědět, co od nich lze čekat a co nikoliv.
- ▶ Programování rozvíjí analytické myšlení, problem-solving skills.
- ▶ Může to být zábavné, jako když jste si (možná) hráli se stavebnicí.
- ▶ Část logiky se dá velmi pěkně formulovat jazykem počítačových programů (Gödelovy věty, ...)

- ▶ Počítač umí provést jakýkoliv proces, který jsme schopni popsat.

Jaké procesy jsme schopni (přesně) popsat?

- ▶ Návrh algoritmů/programů.
- ▶ Analýza algoritmů/programů.
- ▶ Experimentování.

Turingův stroj

<http://www.morphett.info/turing/turing.html>

Aktuální stav	Přečtený znak	Přepsat znakem	Směr posunu	Následující stav
START	Z	Z	R	FFC
FF0	E	E	R	FF0
FF0	1	1	R	FF0
FF0	0	E	R	FFC
FF0	K	K	L	FB0
FB0	E	E	L	FB0
FB0	1	1	L	FB0
FB0	0	E	L	FFC
FB0	Z	Z	R	H1
FF1	E	E	R	FF1
FF1	0	0	R	FF1
FF1	1	E	R	FFC
FF1	K	K	L	FB1
FB1	E	E	L	FB1
FB1	0	0	L	FB1
FB1	1	E	L	FFC
FB1	Z	Z	R	H0
FFC	E	E	R	FFC
FFC	K	K	L	FBC
FFC	0	E	R	FF1
FFC	1	E	R	FF0
FBC	E	E	L	FBC
FBC	Z	Z	R	HE
FBC	0	E	R	FF1
FBC	1	E	R	FF0
H0				
H1				
HE				

Ideální by bylo, moci popsat algoritmus přirozeným jazykem.

- ▶ Počítače (zatím) nerozumí přirozenému jazyku.
- ▶ Přirozený jazyk není jednoznačný.

Viděl jsem s Petrem tři lidi.

- ▶ (A to ani nemluvím o ironii.)
- ▶ Přirozený jazyk má i jiné problémy:

Najdi nejmenší přirozené číslo, které nelze popsat méně jak dvanácti slovy.

Nízkoúrovňové jazyky

- ▶ Strojový kód, Assembler, ...

Jazyky vyšší úrovně (kompilované)

- ▶ C, Fortran ...
- ▶ C++, Pascal, Ada, ...
- ▶ Java

Jazyky vyšší úrovně (interpretované)

- ▶ Basic, Perl, Ruby, Python

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

```
>>> print("Hello, World")
```

```
Hello, World
```

```
>>> print(2 + 3)
```

```
5
```

```
>>> print("2 + 3 =", 2 + 3)
```

```
2 + 3 = 5
```

```
>>>
```

```
>>> def hello():  
    print("Hello")  
    print("Computers are fun")
```

```
>>>
```

Odsazení říká, že řádky patří k funkci hello!

```
>>> hello()  
Hello  
Computers are fun  
>>>
```

K čemu jsou závorky?

Příkazy/Funkce mohou mít měnící se části - parametry.

```
>>> def greet(person):  
    print("Hello", person)  
    print("How are you?")
```

```
>>>
```

```
>>> greet("John")  
Hello John  
How are you?  
>>> greet("Emily")  
Hello Emily  
How are you?  
>>>
```

- ▶ Když zavřeme Pythonní shell, všechny příkazy se nám ztratí!
- ▶ Proto se definice ukládají do souborů, tzv. modulů (nebo skriptů).
- ▶ Tyto soubory jsou obyčejné soubory. Lze je editovat v NotePadu, dokonce, při troše (velké) námahy i ve Wordu.
- ▶ Pro editaci se ale mnohem lépe hodí speciální editory, např. IDLE.

```
# Soubor: chaos.py  
# Jednoduchy program, který ukazuje chaotické  
# chování.  
  
def main():  
    print("Program ukazuje chaotickou funkci:")  
    x = int(input("Zadej číslo mezi 0 a 1: "))  
    for i in range(10):  
        x = 3.9 * x * (1 - x)  
        print(x)  
  
main()
```

Soubor uložíme jako 'chaos.py'.

- ▶ Program lze spustit různými způsoby, např. z příkazové řádky

```
Z:\home>C:\Python34\python.exe chaos.py
Tento program ukazuje chaotickou funkci:
Zadej cislo mezi 0 a 1: 0.25
0.73125
...
```

- ▶ nebo dvojklikem na ploše (nemusí fungovat).
- ▶ Spuštěním pythonu a použitím příkazu `import`, `run` nebo `execute`

```
>>> import chaos
Tento program ukazuje chaotickou funkci:
Zadej cislo mezi 0 a 1: 0.25
0.73125
...
```


- ▶ příkaz `import` řekne pythonu aby program načel do paměti
- ▶ POZOR: přípona `.py` se nezadává!
- ▶ soubor se musí nacházet v adresáři, ve kterém byl python spuštěn
- ▶ při načtení python vytvoří speciální (předkompilovaný) soubor `chaos.pyc`
- ▶ po načtení jsou všechny funkce programu k dispozici pomocí 'tečkového zápisu'

```
>>> chaos.main()
```

Tento program ukazuje chaotickou funkci:

Zadej číslo mezi 0 a 1:

...

První tři řádky začínají znakem #, jsou určeny pro čtenáře, python je ignoruje

```
# Soubor: chaos.py  
# Jednoduchy program, který ukazuje chaotické  
# chování.
```

Pak následuje definice funkce s názvem `main`:

```
def main():
```

Technicky vzato to nebylo nutné, avšak tradičně se instrukce, ze kterých je program složen, vkládají do funkce zvané `main`. Výhodou je, že lze program znovu spustit pomocí tečkové notace: `chaos.main()`

Další řádka je vlastně začátkem našeho programu. Vypíše kratičkou zprávu na obrazovku:

```
print("Program ukazuje chaotickou funkci:")  
x = input("Zadej cislo mezi 0 a 1: 0.25: ")
```

- ▶ x je příkladem proměnné
- ▶ Proměnné se používají, když si chceme označit nějakou hodnotu, abychom se na ni mohli později odkázat.
- ▶ Python vypíše Zadej cislo mezi 0 a 1: a čeká na uživatele
- ▶ To, co uživatel zadá je uloženo jako proměnná x

```
for i in range(10):
```

- ▶ Tato řádka je příkladem cyklu.
- ▶ Cyklus je způsob, jak říci Pythonu, aby něco vícekrát opakoval.
- ▶ Tento konkrétní cyklus říká, že se následující odsazené řádky mají zopakovat 10x:

```
x = 3.9 * x * (1 - x)  
print(x)
```

Stejného efektu bychom docílili, kdybychom tyto řádky napsali 10x pod sebe:

```
x = 3.9 * x * (1 - x)  
print(x)  
x = 3.9 * x * (1 - x)  
print(x)  
... (8x)
```

```
x = 3.9 * x * (1 - x)
```

- ▶ Tento řádek je příkladem **přiřazení**.
- ▶ Na pravé straně '=' je matematický výraz: $3.9 * x * (1-x)$
- ▶ Pokud hodnota x , kterou uživatel zadal, byla např. 0.25, vidíme, že tento výraz má hodnotu $3.9 \times 0.25 \times (1 - 0.25)$, což je 0.73125.
- ▶ Tato hodnota je nyní přiřazena (uložena) do proměnné, která je nalevo od '=', v našem případě x .
- ▶ Nová hodnota x (0.73125) nahradí starou hodnotu (0.25).

Co udělá další řádek už víme:

```
print(x)
```

Vypíše aktuální hodnotu proměnné x , tedy 0.73125. První číslo, které program vypíše, je tedy 0.73125.

Tento proces se ale ještě 9x zopakuje. Avšak hodnota x se stále mění. Například při druhém průchodu, bude na pravo od '=' matematický výraz $3.9 \times 0.73125 \times (1 - 0.73125)$, což je 0.76644140625.

- ▶ Program počítá speciální příklad tzv. **logistické funkce**.
- ▶ Tato funkce má obecný tvar:

$$k \times x \times (1 - x)$$

- ▶ Modeluje chování některých nestabilních el. obvodů nebo populací za omezujících podmínek.
- ▶ Posloupnost je zdánlivě nepředvídatelná.
- ▶ Je citlivá na počáteční hodnotu - efekt motýlích křídel.
- ▶ !!! Chaos \neq Náhodnost !!!
- ▶ Spojitost s dynamickými systémy.

www.python.org

- ▶ Sekce download
- ▶ Je třeba instalovat verzi 3.něco

<https://www.python.org/ftp/python/3.5.1/python-3.5.1-amd64.exe>