

Psaní jednoduchých programů
(2. přednáška)



FF UK

Department of Logic

Jonathan L. Verner

- ▶ Vývoj není jednoduchý.
- ▶ Počítači je nutné říci všechno do nejmenších detailů.
- ▶ Počítač je tupý.
- ▶ Proto je třeba **systematického** přístupu.

- ▶ Rozbor problému problém je třeba co nejpřesněji formulovat, stanovit priority, ujasnit si, kdo bude uživatelem, ...
- ▶ Specifikace programu přesně stanovit, co bude program dělat (i co dělat nebude), jaké budou vstupy/výstupy. V této fázi se není třeba starat o to jak ale o to co.
- ▶ Návrh programu navrhnout algoritmy (datové struktury) strukturu programu. Víme už co, teď jde o to jak. (Architekt)
- ▶ Implementace napsat reálný kód, třeba v Pythonu. (Stavař)
- ▶ Testování/Debugování hlavní otázka: Dělá program opravdu to, co má dělat? Jak se chová, když je uživatel "blbý"? (Uživatelé jsou blbí!) (unit testing, randomized testing...)
- ▶ Údržba postupný vývoj (nové funkce, změny, ...)

Kamarád je na studijním pobytu v USA a dělá mu potíže zvyknout si na měření teploty ve Farenheitech. Anglicky umí dobře, takže ranní předpověď počasí z radia si poslechne, ale moc mu to nepomůže, protože netuší, jestli 32 Farenheitů je zima nebo vedro. Chcete se předvést, tak mu navrhnete, že napíšete program, který mu pomůže.

Tento konkrétní problém je jednoduchý.

- ▶ Jde o převod mezi stupnicemi.
- ▶ Přesněji, jde o převod ze stupnice Farenheit do stupnice Celsius.

- ▶ Program uživatele vyzve, aby zadal teplotu ve stupních Farenheit.
- ▶ Pak vypíše odpovídající teplotu ve stupních Celsia.

- ▶ Víme, že $100^{\circ}\text{C} = 212^{\circ}\text{F}$ a $0^{\circ}\text{C} = 32^{\circ}\text{F}$.
- ▶ Také víme, že obě stupnice jsou na sobě lineárně závislé.
- ▶ Tedy: $212 - 32 = 180$ stupňů F. odpovídá $100 - 0 = 100$ stupňům C., tedy 1 stupeň F odpovídá $100/180 = 5/9$ stupňům C.,

$$C = 5/9F + k$$

- ▶ Dosazením $32^{\circ}\text{F} = 0^{\circ}\text{C}$ zjistíme, že $k \approx -17.7$.

1. Načti vstup od uživatele do F
2. Spočti C podle vzorečku $C = 5/9 * F - 17.7$
3. Vypiš na obrazovku C

```
# fahrenheit2celsius.py  
# Program na převod z Farenheit do Celsius  
# autor: Student Logiky  
  
def main():  
    F = float(input("Jaka je teplota ve Farenheitech? "))  
    C = 5/9 * F - 17.7  
    print("Teplota ve stupních Celsia je", C)  
  
main()
```



```
>>>
```

```
Jaka je teplota ve Farenheitech? 32
```

```
Teplota ve stupnich Celsia je 0.07777777777778
```

```
>>> main()
```

```
Jaka je teplota ve Farenheitech? 212
```

```
Teplota ve stupnich Celsia je 100.077777778
```

```
>>>
```

- ▶ Jména (podobně jako v přirozeném jazyce) jsou důležitou součástí jazyka.
- ▶ Pojmenováváme programy (`fahrenheit2celsius`), funkce/příkazy (`main`, `print`), hodnoty/proměnné (`C`, `F`)...
- ▶ V programátorském žargonu jsou to tzv. **identifikátory** (identifiers).
- ▶ Každý identifikátor musí začínat písmenem nebo podtržítkem (`_`)
- ▶ Může obsahovat pouze písmena (z anglické abecedy), číslice nebo podtržítka
- ▶ Rozlišují se malá a velká písmena (t.j. `Ahoj` není totéž jako `ahoj`)
- ▶ Je dobré volit jména krátká a výstižná (**NE:**
`PromennaJeJizHodnotaSeCastoMeniAMuzeBytIZaporna,`
`celsius` uchováající stupně ve Farenheitech)

Zabraná jména (reserved words)

and	del	for	is	raise
assert	elif	from	lambda	return
break	else	global	not	try
class	except	if	or	while
continue	exec	import	pass	yield
def	finally	in	True	False
None	nonlocal	as	with	

- ▶ Programy manipulují s daty.
- ▶ Části programů, které vytváří/počítají nová data se nazývají “výrazy”.
- ▶ Nejjednodušším příkladem výrazu jsou literály.

9.5 5.0 10

- ▶ Jméno (identifikátor proměnné) je také výrazem.
- ▶ Když se vyhodnocuje výraz, jména jsou nahrazena příslušnou hodnotou.

```
>>> x=5
>>> x
5
>>> y
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    y
NameError: name 'y' is not defined
>>>
```

Jméno musí být definováno, dříve než lze použít ve výrazu.

- ▶ Složitější výrazy lze nakombinovat z jednodušších pomocí operací.
- ▶ Pro číselné výrazy jsou k dispozici sčítání (+), odčítání (-), násobení (*), dělení (/) a mocnění (**).
- ▶ Priorita operací je stejná jako na ní jsme zvyklí (+, - < *, / < **)
- ▶ Mezery Python ignoruje, ale hodí se kvůli čitelnosti.
- ▶ Pro seskupování výrazů lze použít závorky (pouze kulaté závorky: (,)).

```
simple_expr ::= <literal> | <identifikator_promenne>  
OP         ::= + | - | * | / | **  
expr       ::= <simple_expr> | ( <expr> ) |  
             <expr> <OP> <expr>
```

- ▶ Každý příkaz v Pythonu má přesně danou syntax (formu) a sémantiku (význam)
- ▶ Pro popis syntaxe/sémantiky se hodí speciální “meta-jazyky” (viz předchozí slide).

`<variable> = <expr>`

Význam: Výraz na pravé straně je vyhodnocen a výsledná hodnota je pojmenována (přiřazena do proměnné) identifikátorem na levé straně.

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

`C = 5/9 * F - 17.7`

`x = 5`

- ▶ do jedné proměnné lze přiřadit vícekrát
- ▶ proměnná si vždy ponechá hodnotu posledního přiřazení

```
>>> myVar = 0
```

```
>>> myVar
```

```
0
```

```
>>> myVar = 7
```

```
>>> myVar
```

```
7
```

```
>>> myVar = myVar + 1
```

```
>>> myVar
```

```
8
```

myVar

0

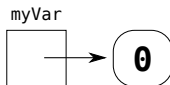
myVar

7

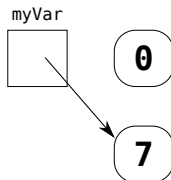
myVar

8

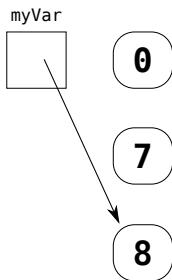
```
>>> myVar = 0  
>>> myVar  
0
```



```
>>> myVar = 0  
>>> myVar  
0  
>>> myVar = 7  
>>> myVar  
7
```



```
>>> myVar = 0
>>> myVar
0
>>> myVar = 7
>>> myVar
7
>>> myVar = myVar + 1
>>> myVar
8
```



- ▶ “Proměnná je ve skutečnosti ukazatel.”
- ▶ Staré hodnoty (na které žádná proměnná neodkazuje) se mažou automaticky “Garbage collection”

```
<variable> = input(<expr>)
```

Význam:

- ▶ Python vyhodnotí výraz `<expr>` a vypíše na obrazovku.
- ▶ Uživatel zadá posloupnost znaků na klávesnici a vstup ukončí klávesou enter.
- ▶ Posloupnost znaků se přiřadí do proměnné `<variable>`.

```
>>> x = input("Zadej cislo: ")
Zadej cislo: 10
>>> x
'10'
>>> x = input("Zadej vyraz: ")
Zadej vyraz: x+10+(3*27)
>>> x
'x+10+(3*27)'
>>>
```

`<variable_1>, ..., <variable_n> = <expr_1>, ..., <expr_n>`

Význam:

- ▶ Python vyhodnotí výrazy na pravé straně,
- ▶ poté výsledky přiřadí do proměnných na levé straně.
- ▶ Hodí se např. k prohození proměnných.

```
>>> x = 10
>>> y = 20
>>> x, y = y, x
>>> print(x, y)
20 10
```

```
for <variable> in <sequence>:  
    <body>
```

Význam:

- ▶ <sequence> je posloupnost hodnot
- ▶ python postupně prochází tuto posloupnost
- ▶ v každém kroku
 - a) přiřadí aktuální hodnotu do proměnné <variable>
 - b) provede příkazy <body>

- ▶ `<sequence>` je často prostě seznamem hodnot
- ▶ o seznamech si více řekneme později, prozatím stačí vědět dva jednoduché způsoby, jak je vytvářet:

```
>>> seznamA=range(2)
>>> for i in seznamA:
    print(i)
```

```
0
1
```

```
>>> seznamB = ["Hynku", "Vileme", "Jarmilo"]
>>> for clovek in seznamB:
    print("Ahoj", clovek, "!")
```

```
Ahoj Hynku !
Ahoj Vileme !
Ahoj Jarmilo !
```

Flowchart diagramy

For cyklus

