Lekce 2

Začínáme programovat I.

Základní konstrukce jazyka: základní datové typy, logické a matematické operátory, podmínky, cykly. Vstup a výstup.

C2184 Úvod do programování v Pythonu
podzim 2014

Mgr. Stanislav Geidl
Národní centrum pro výzkum biomolekul
Masarykova univerzita

Proměnná (variable)

- má svůj název (identifikátor, identifier) `a`
- odkazuje na místo v paměti, kde je uložena tzv. hodnota (value) proměnné
- propojení proměnné a její hodnoty provádíme pomocí přiřazení (assign) (=)

```
a = 1
```
ukázka

- Python je jazyk s dynamickou typovou kontrolou (do proměnné můžete uložit cokoliv, ale hodnoty si pamatují svůj typ)

```
10 + "a"
```
ukázka

- název proměnné se může skládat z malých (a velkých) písmen (bez diakritiky), čísel a podtržítka (_), nesmí začínat číslem, podtžítkem začínají speciální proměnné, název proměnné se nesmí shodovat s žádným z 34 klíčových slov jazyka Python: `and, as, assert, break, class, continue, def, del, elif, else, expect, finally, for, from, global, if, import, in, is, lambda, nonlocal, not, or, pass, raise, return, try, while, with, yield, True, False, None`

Code style - názvy

- `b` (jedno malé písmeno)
- `B` (jedno velké písmeno)
- `lowercase`
- `lower_case_with_underscores`
- `UPPERCASE`
- `UPPER_CASE_WITH_UNDERSCORES`
- `CapitalizedWords` (nebo `CapWords` nebo `CamelCase` nebo `StudlyCaps`)
- `mixedCase`
- `Capitalized_Words_With_Underscores` (hnusné!)

Pro názvy lokálních proměnných budeme používat styl `lowercase` a `lower_case_with_underscores`. Speciální případ bude, když nám název proměnné bude kolidoval s klíčovým slovem např. `class_`.

Code style - přiřazení

Správně:

```
x = 1
y = 2
long_variable = 3
```

Špatně:

```
x=1
y              = 2
long_variable = 3
```


Základní datové typy

1. **bool**ean (booleovský typ) nabývá buď hodnoty `True` nebo `False`.
2. Čísla mohou být celá (**int**eger; 1 a 2), reálná (**float**; 1.1 a 1.2) nebo komplexní (**complex**; 3+1j).
3. Řetězce (**str**ing) jsou posloupnosti Unicode znaků. Tuto podobu může mít například html dokument.
4. Bajty (**byte**) a pole bajtů, například soubor s obrázkem ve formátu jpeg.
5. Seznamy (**list**) jsou uspořádané posloupnosti hodnot.
6. N-tice (**tuple**) jsou uspořádané, neměnné posloupnosti hodnot.
7. Množiny (**set**) jsou neuspořádané kolekce hodnot.
8. Slovníky (**dict**ionary) jsou neuspořádané kolekce dvojic klíč-hodnota.

Odbočka: funkce a jejich použití

- funkce je část programu (je podprogram, angl. subroutine), která lze volat opakovaně z různých části programu, příkladem může být funkce print()

Code style - volání funkce

Správně:

```
# Volání funkce bez mezery
spam(1)
# Vertikální zarovnání na otvirací závorkou
foo = long_function_name(var_one, var_two,
                         var_three, var_four)
# Parametry na dalším řádcích s vlastním
# odsazením
foo = long_function_name(
    var_one, var_two,
    var_three, var_four)
```

Špatně:

```
# Volání funkce s mezery
spam (1)
# Zákaz používat argumenty na prvním řádku,
# pokud nepoužijeme vertikální zarovnání
foo = long_function_name(var_one, var_two,
    var_three, var_four)
```

Zjištění a změna typu

- funkce type()

```
>>> type(10)
<type 'int'>
>>> type(1.5)
<type 'float'>
>>> type(3+1j)
<type 'complex'>
>>> type(True)
<type 'bool'>
>>> type("test")
<type 'str'>
>>> type(None)
<type 'NoneType'>
```

- změna pomocí funkce, která nese název podle typu, který bude výsledkem, např. int()

Zjištění a změna typu

- funkce type()
- změna pomocí funkce, která nese název podle typu, který bude výsledkem, např. int()

```
>>> int("10")
10
>>> int("10a")
ValueError: invalid literal for int()
with base 10: '10a'
>>> int(10.6)
10
>>> float(1)
1.0
>>> bool(0)
False
>>> bool(1)
True
>>> bool(None)
False
>>> str(1)
'1'
```


Matematické operátory

$+$	sčítání (např. $10 + 3$)
$-$	odčítání (např. $10 - 3$)
$*$	násobení (např. $10 * 3$)
$/$	dělení (např. $10/3$)
$//$	celočíselné dělění (např. $10 // 3 = ??$)
%	modulo (zbytek po celočíselném dělení; např. $10 \% 3 = ??$)
$**$	umocňování (např. $10 * *3$)
()	závorky
`abs()`	funkce, která vrací absolutní hodnotu

- pořadí operací jako v matematice
- Co bude výsledkem $2 * *8 - (5 + 5) * 5 * 5 + 5$?

Speciální typy přiřazení: `+=` `-=` `*=` `/=` `%=` `//=`

```
a += 1 # a = a + 1
```


Logické operátory

`and`	a
`or`	nebo
`not`	negace
`==`	je rovno
`!=, <>`	není rovno
`>`	je větší
`<`	je menší
`>=`	je větší rovno
`<=`	je menší rovno
`is, is not`	je, není (např. `1 is int()` nebo `a is not None`)
`in, not in`	je v, není v (např. `'pes' not in 'Harapes'` nebo `'a' in 'test'`)

- pořadí operací: matematické operace; < > <= >=; == !=; <>; is, is not; in, not in; not; and; or

Code style - operátory

Správně:

```
i = i + 1
submitted += 1
x = x*2 - 1
hypot2 = x*x + y*y
c = (a+b) * (a-b)
```

Špatně:

```
i=i+1
submitted +=1
x = x * 2 - 1
hypot2 = x * x + y * y
c = (a + b) * (a - b)
```

Příklad - řešení kvadratické rovnice

Příklad - řešení kvadratické rovnice

$$ax^2 + bx + c = 0 \tag{1}$$

$$D = b^2 - 4ac \tag{2}$$

```
D = b**2 - 4*a*c
```

$$x_{1,2} = \frac{-b \pm \sqrt{D}}{2a} \tag{3}$$

```
x1 = (-b+math.sqrt(D)) / (2*a)
x2 = (-b-math.sqrt(D)) / (2*a)
```

Bloky

- Bloky slouží k seskupení příkazů, například uvnitř cyklů, funkcí, objektů, struktur atd.
- Například v Pascalu jsou bloky označeny slovy BEGIN a END a v jazyce C slouží pro vytváření bloků špičaté závorky . V Pythonu se pro vytváření bloků používá odsazování.
- Nový blok vytvoříte tak, že napíšete na začátku řádků před příkazy, které spolu tvoří blok, libovolný počet mezer nebo tabulátorů. Ale před každým dalším příkazem v bloku musí být stejný počet mezer a tabulátorů! Odsadíte-li o mezeru více, nebo pokud místo tabulátoru použijete mezery, začnete tím jiný blok. Pokud vytvoříte nový blok na místě, kde nemá být, skončí program s chybou.

```
prikaz1:
  blok1
  blok1
prikaz2:
  blok2
  blok2
  prikaz3:
    blok3
prikaz4:
  blok4
```


Podmínky

Na základě vyhodnocení logického výrazu se rozhodne, jestli se daný blok provede nebo nikoliv.

Podmínka if

```
if `podmínka`:
  print("Podmínka splněna.")
```

```
if 9 > 5:
  print("Devět je větší než pět.")
```

```
age = 20
is_man = True
if age >= 18 and is_man:
  print("Plnoletý muž.")
```


Podmínka if .. else

```
if `podmínka`:
  print("Podmínka splněna.")
else:
  print("Podmínka nesplněna.")
```

```
if 9 < 5:
  print("Devět je menší než pět.")
else:
  print("Devět není menší než pět.")
```

```
age = 16
is_man = True
if age >= 18 and is_man:
  print("Plnoletý muž.")
else:
  print("Mladistvím a ženám nic neřeknu.")
```


Podmínka if .. elif .. else

```
if `podmínka` and `podmínka2`:
  print("Podmínka splněna.")
elif `podmínka2`:
  print("Alespoň podmínka2 splněna.")
else:
  print("Podmínky nesplněny.")
```

```
age = 20
is_man = False
if age >= 18 and is_man:
  print("Plnoletý muž.")
elif not is_man:
  print("Ženo, bež pryč.")
else:
  print("Mladistvím nenalévám.")
```


Code style - podmínky

Správně:

```
if not users:
    print 'no users'

if foo == 0:
    self.handle_zero()

if i % 10 == 0:
    self.handle_multiple_of_ten()
```

Špatně:

```
if len(users) == 0:
    print 'no users'

if foo is not None and not foo:
    self.handle_zero()

 if not i % 10:
    self.handle_multiple_of_ten()
```


Příklad - je číslo dělitelné třemi nebo pěti?

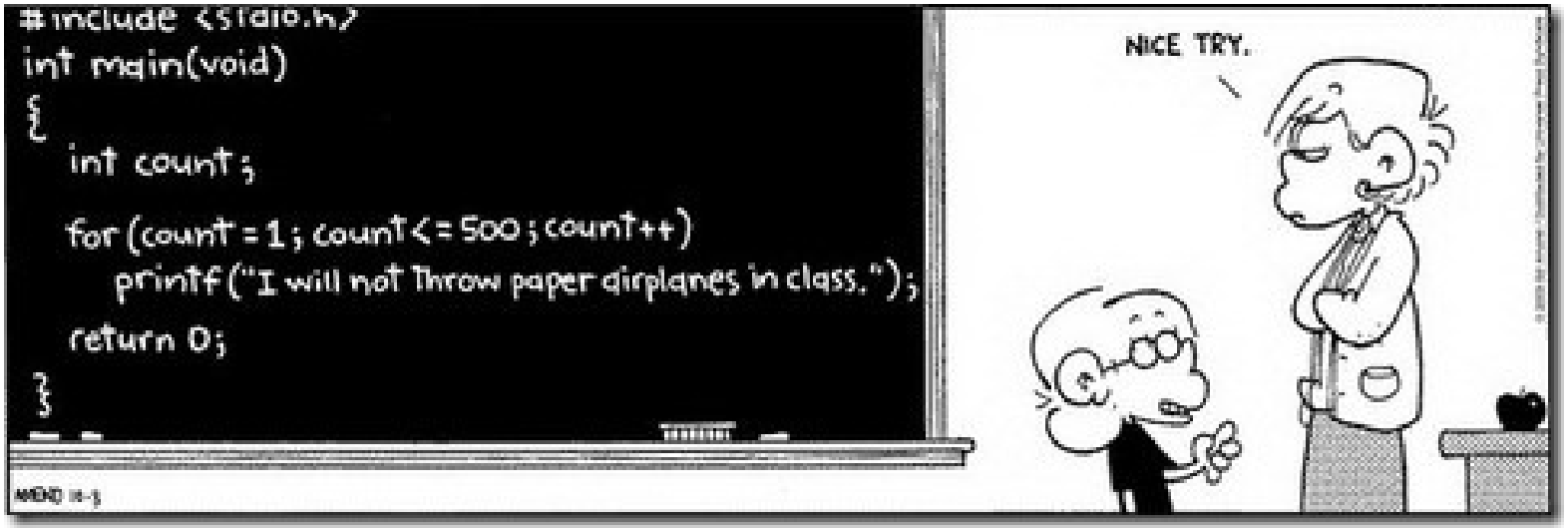
Příklad - je číslo dělitelné třemi nebo pěti?

```
cislo = 14
if cislo%3 == 0 and cislo%5 == 0:
    print("Číslo je dělitelné třemi i pěti.")
elif cislo%3 == 0:
    print("Číslo je dělitelné třemi.")
elif cislo%5 == 0:
    print("Číslo je dělitelné pěti.")
else:
    print("Číslo není dělitelné třemi ani pěti.")
```

ukázka

Cykly

Podobně jako podmínky se řídí logickým výrazem, který rozhoduje o spuštění příslušného bloku, tyto bloky mohou (většinou to i chceme :)) běžet i několikrát po sobě.



Šetří nám čas s psaním kódu a ruce od používání Ctrl+C/V.

Cyklus while

```
while `podminka`:
    blok
    blok

i = 0
while i < 10:
    print(i)
    i += 1
```

ukázka

Cyklus for

```
for i in ...:
    blok
    blok

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

ukázka

Řízení běhu cyklu pomocí break a continue

- `break` vynutí ukončení cyklu
- `continue` vynutí ukončení vykonávání bloku a spustí další smyčku

```
for i in range(100):
    if i%3 == 0:
        print("<3")
        continue
    if i >= 10:
        break
    print(i)
```

ukázka

Řízení běhu cyklu pomocí break a else

- `else` pokud cyklus nebyl ukončen pomocí `break` spustí svůj blok

```
for i in range(1,10):
    print(i)
    if i % 11 == 0:
        break
else:
    print("Žádné číslo dělitelné 11 nenalezeno.")
```

ukázka

Příklad - výpočet odmocniny pomocí Newtonovy metody

Tato metoda hledá řešení rovnice $f(x) = 0$ a odhaduje (zlepšuje odhad) na základě tohoto výpočtu:

$$x_{betterguess} = x_{guess} - \frac{f(x_{guess})}{f'(x_{guess})} \tag{4}$$

tento výpočet je neustále iterován, dokud nejsme s výsledkem spokojeni.

Příklad - výpočet odmocniny pomocí Newtonovy metody

$$x = \sqrt{a} \tag{5}$$

$$f(x) = x^2 - a = 0 \tag{6}$$

$$f'(x) = 2x \tag{7}$$

$$x_{betterguess} = x - \frac{x^2 - a}{2x} = \frac{x + \frac{a}{x}}{2} \tag{8}$$

```
a = 9
x = 1.0 # initial guess
while abs(x**2-a) > 0.000000001:
    x = (x+a/x) / 2
    print(x)
```

ukázka

Vstupy a výstupy

Vstup:

- `raw_input()` v Pythonu 2.x nebo `input` v Pythonu 3
- argumenty programu (přes modul `sys`):

```
import sys
if len(sys.args) > 1:
    print sys.argv[1]
```

- ze souboru

Výstup:

- print()
- `sys.stdout` a `sys.stderr`
- do souboru

Domací úkoly

1. Stáhněte si ze studijních materiálů `homework02A_template.py` a do vyznačené části (mezi `## ZACATEK ##` a `## KONEC ##`) s použitím předdefinovaných proměnných (`R`, `e0` (=ϵ_0), ...) dopište program pro řešení vybrané rovnice (**vybíráte pouze jednu, 4 body**).
2. Napište program, který na výstup vypíše prvních 15 prvků vybrané posloupnosti (**vybíráte pouze jednu, 4 body**).

- První úkol do odevzdávárny nahrajte jako `homework02A.py` a druhý jako `homework02B.py`, vaše jméno a UČO se přidá automaticky.
- Nevytvářejte složky v odevzdávárně prosím.
- **Nejpozději do 30. 9. 2014 včetně!**