Lekce 3

Podmínky a cykly.

Logické operátory, podmínky, cykly.

C2184 Úvod do programování v Pythonu
podzim 2016

Stanislav Geidl
Národní centrum pro výzkum biomolekul
Masarykova univerzita

Logické operátory

`==`	je rovno
`!=`, `<>`	není rovno
`>`	je větší
`<`	je menší
`>=`	je větší rovno
`<=`	je menší rovno
`and`	a
`or`	nebo
`not`	negace
`in`, `not in`	je v, není v (např. `'pes' not in 'Harapes'` nebo `'a' in 'test'`)
`is`, `is not`	je, není (testování identity objektů, nezáleží na hodnotě ale na "místě v paměti)

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

Logické operátory

`and`	a
`or`	nebo
`not`	negace

a	b	a and b	a or b	not a
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

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.")


x = 5
if x >= 2 and x < 10:
    print("Proměnná náleží do intervalu <2;10)")
```


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.")


x = 13
if x >= 2 and x < 10:
    print("Proměnná náleží do intervalu <2;10)")
else:
    print("Proměnná nenáleží do intervalu <2;10)")
```


Podmínka if .. elif .. else

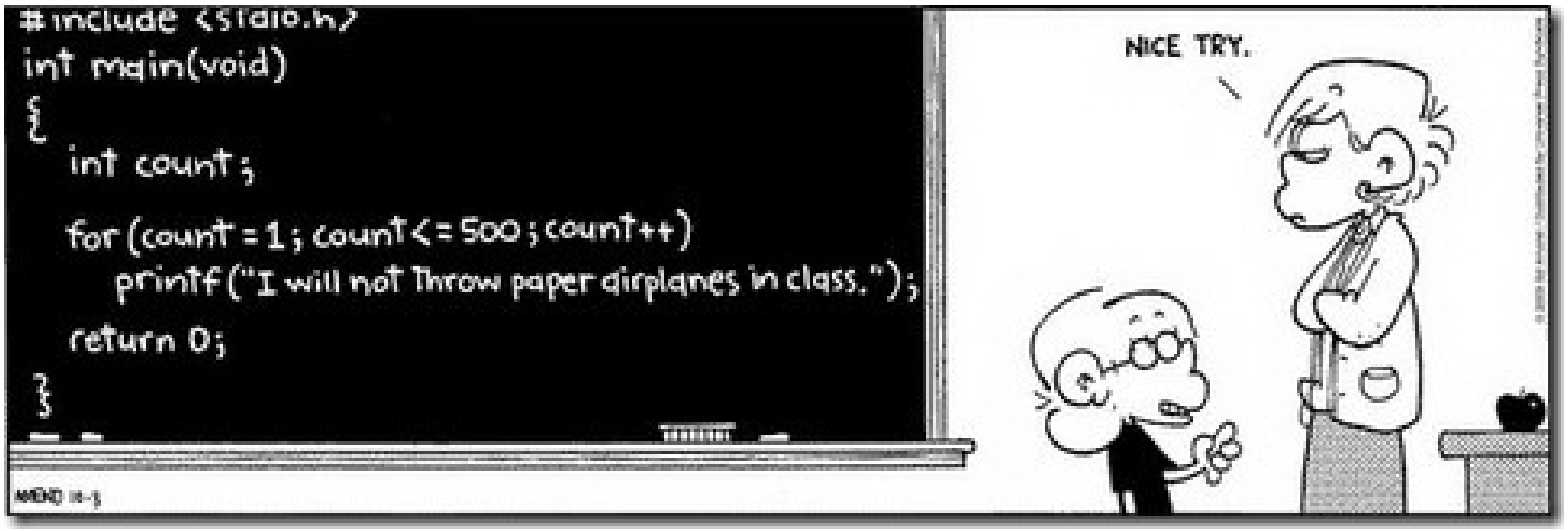
```
if `podmínka` and `podmínka2`:
    print("Podmínky splněny.")
elif `podmínka`:
    print("Alespoň podmínka splněna.")
elif `podmínka2`:
    print("Alespoň podmínka2 splněna.")
else:
    print("Podmínky nesplněny.")
```

```
x = 11
if x >= 2 and x < 10:
    print("Proměnná náleží do intervalu <2;10)")
elif x >= 5 and x < 15:
    print("Proměnná náleží do intervalu <5;15)")
else:
    print("Proměnná x mimo povolený rozsah!")
```

Kdyby *x* bylo rovno 8, který blok by se provedl?

Cykly

Podobně jako podmínky se řídí logickým výrazem, který rozhoduje o spuštění příslušného bloku. Tyto bloky běží stále dokola, dokud je splněna podmínka.



Cyklus while

```
while `podminka`:
    blok
    blok

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

Proměnné řídící běh cyklu se často pojmenovávají `i, j, k, l,...`, pokud nemají jiný význam.

Cyklus for

- určen pro procházení kolekcí či jiných objektů, které můžeme procházet (případně řetězců)

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

- funkce range umožňuje cyklu for procházet číslné posloupnosti podobně jako cyklu while

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


Funkce range

- rozsah **PO** `end` (od 0, po kroku 1)

```
range(end)
range(10)
```

- rozsah od `start` PO `end` (po kroku 1)

```
range(start, end)
range(1, 100)
range(-50, 0)
```

- rozsah OD `start` PO `end` po kroku `step`

```
range(start, end, step)
range(10, 51, 50)
range(0, -10, -1)
```

- zkuste

```
for i in range(0, -10): print(i)
for i in range(0, 10, 3): print(i)
```


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)
```


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.")
```


Input a isinstance()

- funkce input()

Python 2	Python 3
Získání dat ve formě řetezce:	
raw_input()	input()
Získání dat automaticky převedených na konkretní typ:	
input()	eval(input()) [1]

- funkce isinstance() pro testování, jestli se jedná o číslo, ...

```
vstup = eval(input())
if isinstance(vstup,int):
    print("Vstup je celé číslo.")
```

[1] používání funkce eval je nebezpečné, viz http://nedbatchelder.com/blog/201206/eval_really_is_dangerous.html

Příklady I.

1. Pomocí funkce input získejte řetězec, ověrte jestli se jedná o celé kladné číslo (řetězec se skládá pouze z číslic) a převeďte jej pomocí funkce int() na číslo `a`.
2. Zjistěte, jestli je číslo `a` dělitelné 3 a 5 nebo pouze 3 nebo pouze 5 nebo číslo není dělitelné ani jedním.
3. Zjistěte, jestli je číslo `a` menší rovno 10 a pokud ano spočítejte faktoriál, pokud je číslo větší informujte uživatele, že číslo je příliš velké.

Příklady I. - řešení

```
a = input("Zadejte celé kladné číslo: ")
if a.isdigit():
    a = int(a)
else:
    print("Nejdná se o číslo.")
if a%3 == 0 and a%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.")
if a <= 10:
    faktorial = 1
    for i in range(2,a+1):
        faktorial *= i
    print(faktorial)
else:
    print("Číslo je příliš velké.")
```


Příklady II.

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

$$x_{betterguess} = x_{guess} - \frac{f(x_{guess})}{f'(x_{guess})} \tag{1}$$

$$x = \sqrt{a} \tag{2}$$

$$f(x) = x^2 - a = 0 \tag{3}$$

$$f'(x) = 2x \tag{4}$$

$$x_{betterguess} = x - \frac{x^2 - a}{2x} = \frac{x + \frac{a}{x}}{2} \tag{5}$$

tento výpočet je několikrát iterován nebo je iterován, dokud nejsme s výsledkem spokojeni.

Příklady II. - řešení


```
"""
hodnota x bude nastavena na nějaký odhad
hodnota threshold bude rozhodovat o kvalitě výsledku
"""
x = 1.0
threshold = 0.000000001
a = input("Zadejte celé kladné číslo: ")
if a.isdigit():
    a = int(a)
    while abs(x**2-a) > threshold:
        x = (x+a/x) / 2
        print(x)
""" alternativní výpočet, který provede přesně 500 iterací
    x = 1.0
    for i in range(500):
        x = (x+a/x) / 2
        print(x)
```