Programovanie v jazyku Python

Syntax, základné jazykové konštrukty, vývojové diagramy

prednáška 1

Katedra kybernetiky a umelej inteligencie
Technická univerzita v Košiciach
Ing. Ján Magyar

História Pythona

- myšlienka pochádza z 80-tych rokov - Guido van Rossum
- 1990 - Python 1.0
- 2000 - Python 2.0
- 2008 - Python 3.0
- súčasne - Python 2.7.X a Python 3.9.X

Vlastnosti Pythona

- všeobecne použiteľný
- vyšší
- interpretovaný
- viacparadigmový
  - štruktúrované/procedurálne programovanie
  - objektovo orientované programovanie
  - funkcionálne programovanie
  - čiastočne aspektovo orientované programovanie
  - čiastočne metaprogramovanie
  - cez rozšírenia logické programovanie

Python vs. C

- interpretovaný
- vyšší programovací jazyk
- viacparadigmový
- dynamická typová kontrola
- premenné, garbage collection
- podpora definície údajových štruktúr
- syntakticky významné odsadenie

- kompilovaný
- stredný programovací jazyk
- procedurálne programovanie
- statická typová kontrola
- smerníky, alokácia pamäte
- explicitná definícia údajových štruktúr
- bloky pomocou zátvoriek

Základné jazykové konštrukty - prehľad

- hodnoty a premenné
- operátory
- vetvenie
- iterácie
- funkcie

Premenné a hodnoty

- dynamická typová kontrola
- definícia premennej v C:

```
typ názov = hodnota;
```

- definícia premennej v Pythone:

```
názov = hodnota
```

Názvy premenných

„There are only two hard things in Computer Science: cache invalidation and naming things.“

-- Phil Karlton

- kľúčové slová sú zakázané
- nemali by ste používať názvy štandardných metód a funkcií
- nemali by ste používať písmená `O`, `I`, `l`
- musia začať na písmeno
- čo najkratšie ale zrozumiteľné

Zvyky pri pomenúvaní

- premenné, metódy a funkcie: malé začiatočné písmená, slová oddelené _

  `my_wonderful_variable`, `my_wonderful_function`
- triedy: veľké začiatočné písmená, camelcase

  `MyClass`
- konštanty: veľké písmená, slová oddelené _

  `MY_CONSTANT`
- moduly: malé začiatočné písmená, slová oddelené _

  `my_module`
- balíky: malé začiatočné písmená, slová písané spolu

  `mypackage`

Primitívne typy v Pythone

- integer
- float
- complex (napr. `3 + 4j`)
- boolean (`True` alebo `False`)
- string (napr. `'abc'` alebo `"abc"`)
- None

Sekvenčné typy v Pythone

- list (zoznam)
  - meniteľný
  - postupnosť hodnôt rôzneho typu (zvyčajne ale homogénna)
  - `[1, 2.4, 'abc']`
- tuple (n-tica)
  - nemeniteľný
  - postupnosť hodnôt rôzneho typu (zvyčajne heterogénna)
  - `(1, 2.4, 'abc')`
- range (interval)
  - nemeniteľný
  - obsahuje čísla
  - tri parametre: start, stop, step
  - `range(3, 8)`

Mapovacie typy v Pythone

dictionary (asociatívne pole )

- mapuje hašovateľné hodnoty na ľubovoľné hodnoty
- skladá sa z dvojíc kľúč-hodnota
- kľúčom nemôže byť: zoznam, dictionary, meniteľné hodnoty
- `dct = {'boys': ['', '', ''], 'girls': ['', '']}`

Množinové typy v Pythone

- nezoradená množina jedinečných hašovateľných hodnôt
- používa sa pre:
  - určenie príslušnosti
  - vymazanie duplikátov
  - množinové operácie
- set
  - meniteľný
  - nehašovateľný
  - `{'ab', 'bc'}`
- frozenset
  - nemeniteľný
  - hašovateľný

(Binárne typy v Pythone)

- bytes
  - nemeniteľné
  - postupnosť byteov
- bytearray
  - meniteľný
- memoryview
  - umožňuje prístup do pamäte
  - smerník na pamäť kde sa nachádza časť objektu
  - iba pre objekty podporujúce buffer protocol

Operátory v Pythone

- aritmetické operátory
- priraďovacie operátory
- porovnávacie operátory
- logické operátory
- operátory identity
- operátory príslušnosti
- bitové operátory

Aritmetické operátory v Pythone

+	sčítanie
-	odčítanie
*	násobenie
/	delenie
%	modulo (zvyšok)
//	celočíselné delenie
**	umocňovanie

Prirad'ovacie operátory v Pythone

=	x = 5	&=	x = x & 5
+=	x = x + 5	|=	x = x | 5
-=	x = x - 5	^=	x = x ^ 5
*=	x = x * 5	>>=	x = x >> 5
/=	x = x / 5	<<=	x = x << 5
%=	x = x % 5		
//=	x = x // 5		
**=	x = x ** 5		

Walrus operátor

(x:=5)

- od Python 3.8
- priradí hodnotu a vráti ju
- kritizovaný
  - pre každú operáciu by mal existovať iba jeden operátor
  - jednoduchosť je lepšia ako komplexita
  - nikto nevie, ako ho vývojári budú používať

Používanie walrusa

```
# f() may return None
x = f()
if x:
    process(x)
```

```
# f() may return None
if (x:=f()):
    process(x)
```

```
[f(x), f(x)**2, f(x)**3]
```

```
[y:=f(x), y**2, y**3]
```

Porovnávacie operátory v Pythone

==	rovná sa
!=	nerovná sa
>	väčšie
<	menšie
>=	väčšie alebo rovné
<=	menšie alebo rovné

Logické operátory v Pythone

`and`	zároveň
`or`	alebo
`not`	nie je

Operátory identity v Pythone

`is`	je
`is not`	nie je

Operátory príslušnosti v Pythone

`in`	nachádza sa
`not in`	nenachádza sa

(Bitové operátory v Pythone)

&	bitový AND (bit je 1 ak oba bity sú 1)
|	bitový OR (bit je 1 ak jeden z bitov je 1)
^	bitový XOR (bit je 1 ak bity sú odlišné)
~	bitový NOT (invertuje všetky bity)
<<	left shift (posúva bity doľava, pridáva nuly)
>>	right shift (posúva bity doprava, pridáva nuly)

Vetvenie - podmieňovací príkaz

```
if podmienka:
        telo
elif podmienka:
        telo
else:
        telo
```

Iterácie - cykly

všeobecne tri typy:

1. aritmetický
    - for
2. logické
    - **while**
    - do … while
3. **foreach**

Logické cykly v Pythone

```
while podmienka:
        telo
```

do … while v Pythone

```
telo
while podmienka:
        telo
```

Foreach cyklus v Pythone

- pre iteráciu nad prvkami sekvencie

  for e in sequence:
  (do something with e)

- sekvencia môže byť:
  - zoznam
  - n-tica
  - interval (range)
  - množina (set/frozenset)
  - string - prvky sú znaky

Aritmetický cyklus v Pythone

- jazyky založené na C obsahujú cyklus `for` v tvare:

```
for (int i = 0; i < 5; i++) { telo; }
```

- reprezentácia v Pythone

```
for i in range(0, 5):
    telo
```

- aktualizácia počítadla je možná pomocou parametra step

Vývojové diagramy

- grafická reprezentácia krokov algoritmu, resp. procesu
- každý krok je reprezentovaný blokom, ktoré sú prepojené šípkami
- šípky smerujú zhora dole a sprava doľava
- nezávislé od použitého programovacieho jazyka

Terminálne symboly

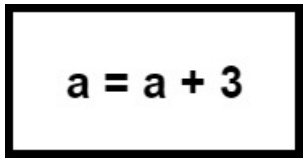
- vyjadrujú začiatok a koniec algoritmu
- musí ich obsahovať každý vývojový diagram

- START

- END

Proces

- vyjadruje súbor operácií, ktoré menia hodnoty premenných a údajov



Rozhodovanie

- vyjadruje podmieňovací príkaz
- určuje, ktorou z dvoch vetiev bude pokračovať program
- zvyčajne otázka typu áno/nie, alebo test pre zistenie pravdivosti
- používa sa pri vetveniach a cykloch

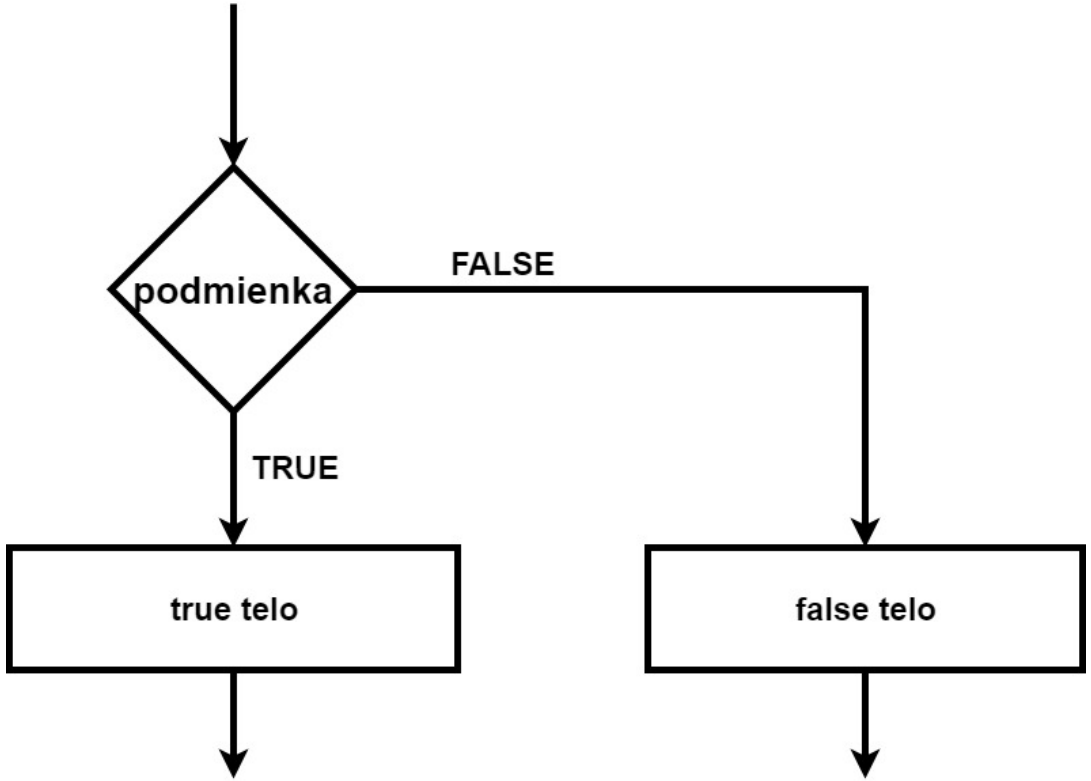
Vstupno-výstupné operácie

- vyjadruje proces získania alebo výpisu údajov
- iba explicitné vstupy a výstupy (parameter funkcie a návratová hodnota nie)

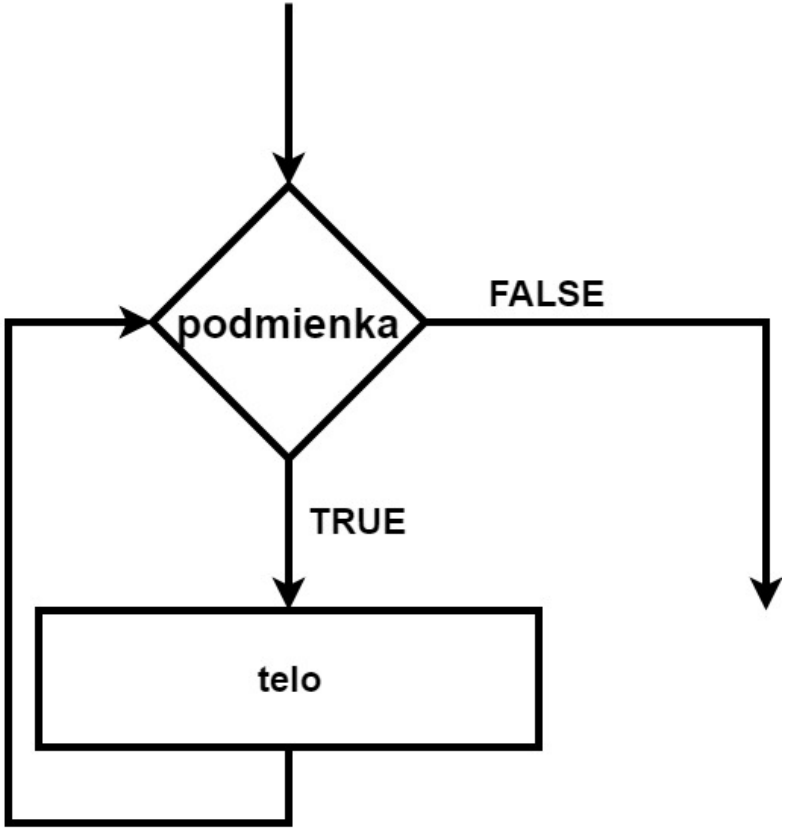
Preddefinované procesy

- pomenovaný proces, ktorý už bol zdokumentovaný
- volanie vlastných funkcií

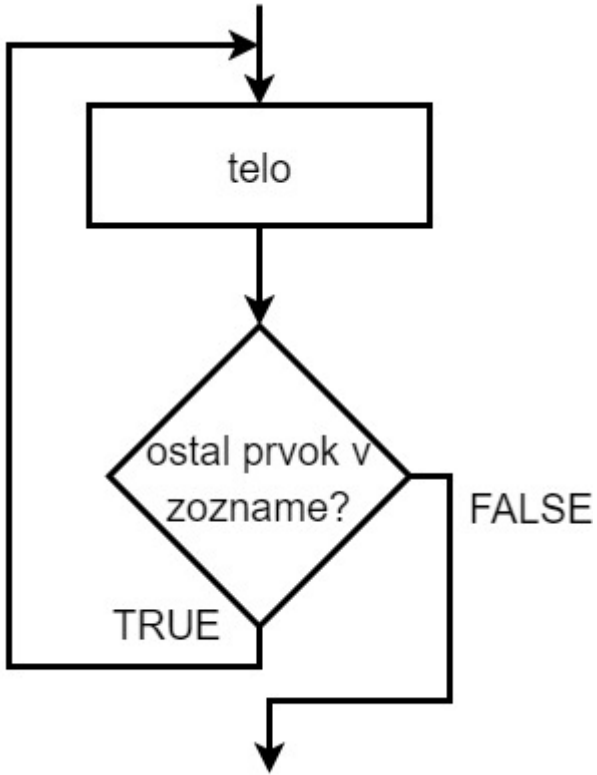
Reprezentácia vetvenia



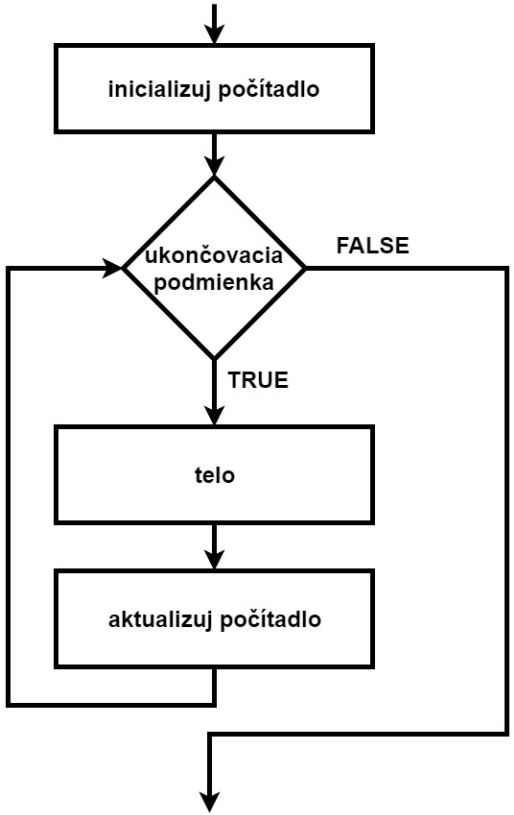
Reprezentácia while cyklov



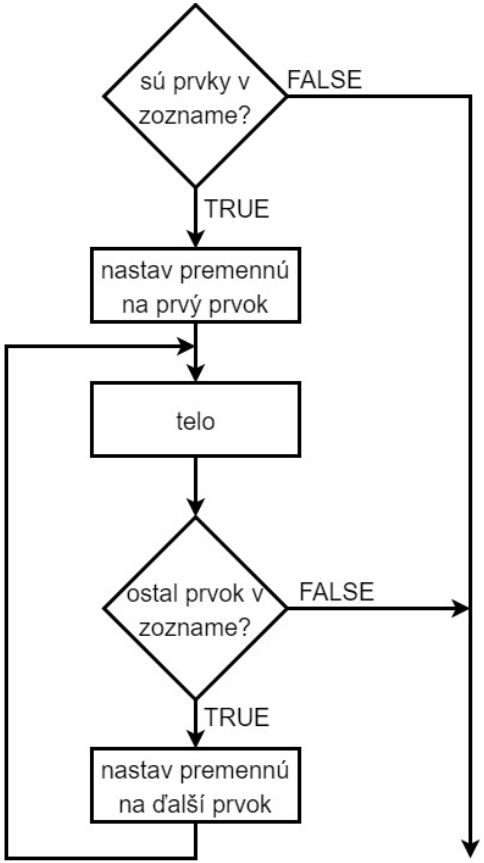
Reprezentácia do ... while cyklov



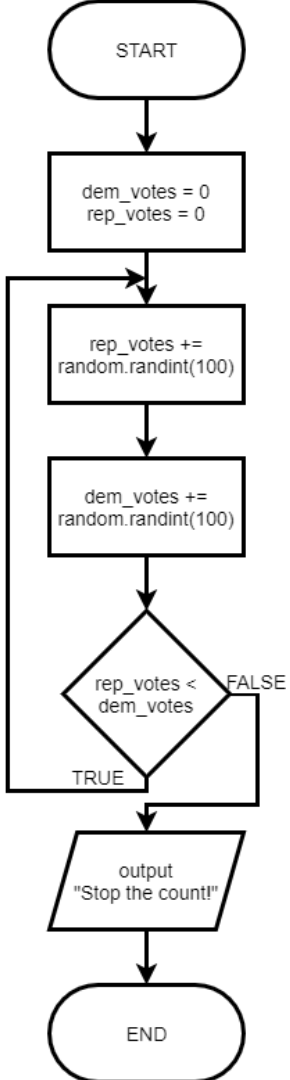
Reprezentácia for cyklov



Reprezentácia foreach cyklov



Ukážka


START
dem_votes = 0
rep_votes = 0
rep_votes +=
random.randint(100)
dem_votes +=
random.randint(100)
rep_votes <
dem_votes
FALSE
TRUE
output
"Stop the count!"
END

Zhrnutie

- základné vlastnosti Pythonu
- rozdelenie základných konštruktov
- rozdelenie operátorov
- vetvenia a cykly
- vývojové diagramy