Cykly

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/ZPP1 Základy programování v Pythonu 1

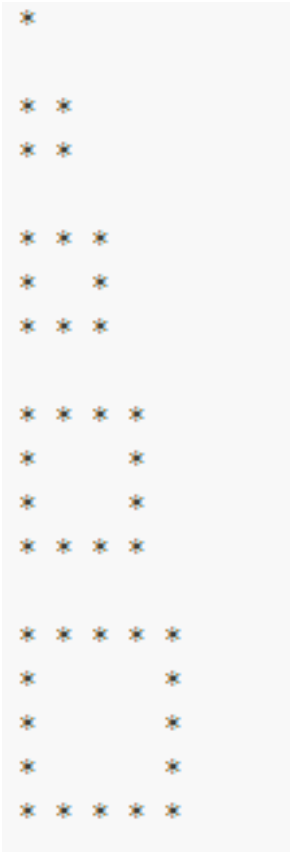
Řešení úkolu z minulé hodiny

```
import math
a=1
b=7
c=12

D=b**2-4*a*c
if(D<0):
    print("Úloha nemá řešení v oboru reálných čísel.")
elif (D==0):
    x=-b/(2*a)
    print("Rovnice má dva stejné reálné kořeny",x,".")
else:
    x1=(-b+math.sqrt(D))/2*a
    x2=(-b-math.sqrt(D))/2*a
    print("Rovnice má dva  reálné kořeny",x1,"a",x2,".")
```

Zadání 1. příkladu

- Napište program, který pro zadané číslo n postupně vytiskne na obrazovku čtverce, které budou mít i řádků a i sloupců (i=1,…,n). Maximální hodnota n bude 11 (součástí programu musí být i test přípustnosti n) a minimální 1.
- Výstup pro n=5:



Podmínky vypracování příkladu

- Za každý příklad může student získat **0-4 body**.
- Body se strhávají v těchto případech:
  - Řešení je **nekompletní**; byla opomenuta nějaká část, mezní hodnoty vstupů apod.
  - Student odevzdával velmi **neodladěné** řešení, i když jej poté opravil.
  - Řešení je sice funkční, ale program vypadá obzvláště **odpudivě**.
  - Student vypracoval své řešení s **pomocí** jiného studenta a tuto spolupráce předem oznámil.
  - Student odevzdal řešení **po termínu**.
- Řešení student odevzdá elektronicky do **31.10.2023**.

Cykly

Příklad

- Vypište prvních 5 přirozených čísel

```
print("1")
print("2")
print("3")
print("4")
print("5")
```

- Lze úkol splnit tak, aby příkaz tisku byl při každém kroku stejný?

```
i=1
print(i)
i=2
print(i)
i=3
print(i)
i=4
print(i)
i=5
print(i)
```

Cyklus FOR

Cyklus for

- syntaxe:

```
for parametry_cyklu in zdroj: příkaz

for parametry_cyklu in zdroj:
    příkazy
```

- co může být zdrojem:
  - n-tice,
  - číselná posloupnost,
  - libovolný iterovatelný objekt.

n-tice

- jsou proměnné posloupnosti (instance třídy tuple)
- Vytvoření:

```
ntice=1,2
ntice=(1,2)
ntice=(1, "P", 120, "k")
```

Příklad

- Vypište prvních 5 přirozených čísel:

```
for i in (1,2,3,4,5):
    print (i)
```

- Vypište prvních n přirozených čísel?

Funkce `range`

- Slouží k vytvoření sekvence celých čísel.
- Syntaxe

```
range(start, stop, krok)
```

  - nepovinný argument start určuje první číslo sekvence (není-li uveden, je první číslo 0),
  - povinný argument stop určuje poslední číslo sekvence, které již v sekvenci není,
  - nepovinný argument krok ovlivňuje výpočet následujícího čísla v sekvenci.
- Příklady:

```
range(10) # sekvence: 0, 1, 2, ..., 9
range(1, 11) # sekvence: 1, 2, 3, ..., 10
range(0, 10, 2) # sekvence: 0, 2, 4, 6, 8
range(0, 5, 10) # sekvence: 0
range(0, 0) # prázdná sekvence
range(0) # prázdná sekvence
range(10, 0, -2) # 10, 8, 6, 4, 2
range(-10, 0) # -10, -9, -8, ..., -1
```

Příklad

- Vypište prvních n přirozených čísel

```
n=20

for i in range(1,n+1):
    print (i)
```

Příklad

- Postupně si vyzkoušíme vytisknout jednotlivé sekvence:

```
for i in range(10):
    print (i)
for i in range(1, 11):
    print (i)
for i in range(0, 10, 2):
    print (i)
for i in range(0, 5, 10):
    print (i)
for i in range(0, 0):
    print (i)
for i in range(0):
    print (i)
for i in range(10, 0, -2):
    print (i)
for i in range(-10, 0):
    print (i)
```

Příklad

- Všimnete si, že proměnná i je nastavena před každým vykonáním těla cyklu. Její změnou tedy nelze ovlivnit počet opakování.

```
for i in range(10):
    print(i)
    i = i + 1
```

- Po skončení cyklu bude mít proměnná vazbu na číslo poslední iterace:

```
i = 5
for i in range(10):
    print(i)
print(i)
```

Příklad

- Co když budeme chtít vytisknout všechna sudá čísla menší nebo rovno než zadané číslo?
- První způsob:

```
n = 10
for i in range(1,n+1):
    if i % 2 == 0:
        print(i)
```

- Druhý způsob:

```
n=10
for i in range(1,(n+2)//2):
    print(2*i)
```

- Třetí způsob:

```
n=10
for i in range(2,n+1,2):
    print(i)
```

Příklad

- Následující program vytiskne všechny dělitele zadaného čísla.

```
n = 100

for i in range(1,n+1):
    if n % i == 0:
        print(i)
```

- Předchozí program stačí mírné upravit a obdržíme program rozhodující o tom, zda je dané číslo prvočíslem.

```
n = 7
pocet_delitelu = 0
for i in range(1,n+1):
    if n % i == 0:
        pocet_delitelu = pocet_delitelu + 1

je_prvocislo = pocet_delitelu == 2
print(je_prvocislo)
```

Příklad

- Algoritmus ještě upravíme takto:

```
n = 7
je_prvocislo = True
for i in range(2,n):
    if n % i == 0:
        je_prvocislo= False
print(je_prvocislo)
```

Příklad

- Napíšeme program pro zakódování znaku do morseovky:

```
z='b'
morseovka=(('a','.-'),('b','-...'),('c','-.-.'),('d','-..'),('e','.'))
for znak,kod in morseovka:
    if z==znak: print (kod)
```

- Pokud znak najdeme, je zbytečné v cyklu pokračovat.

Příkazy přerušení cyklu

- příkaz `continue`
  - skok na konec nejvnitřnějšího cyklu, výpočet pokračuje další iterací (včetně testu případné podmínky)
- příkaz `break`
  - okamžité opuštění nejvnitřnějšího cyklu
- Vztahují se vždy k „nejbližšímu“ cyklu

Příklad

- Předchozí program přepíšeme s přerušením:

```
z='b'
morseovka=(('a','.-'),('b','-...'),('c','-.-.'),('d','-..'),('e','.'))
for znak,kod in morseovka:
    if z==znak:
        print (kod)
        break
```

Větev `else`

- cyklus může mít větev `else`,
- příkazy této větve se provedou v případě, že program úspěšně projde celým cyklem,
- bude-li předčasně ukončen příkazem `break`, větev se neprovede

Příklad

- Pokud nám stačí určit, že číslo je prvočíslo, můžeme algoritmus přepsat takto:

```
for i in range(2,n):
    if n%i==0:
        break
else:
    print("Je prvočíslo.")
```

Vnořování cyklů

Příklad – Malá násobilka

```
print("Malá násobilka pomocí cyklu:\n")
for i in range(1,11):
    print(f"\t{i}",end=" ")
print("")
for i in range(1,11):
    print(f"\t{2*i}",end=" ")
print("")
for i in range(1,11):
    print(f"\t{3*i}",end=" ")
print("")
for i in range(1,11):
    print(f"\t{4*i}",end=" ")
print("")
for i in range(1,11):
    print(f"\t{5*i}",end=" ")
print("")
…
```

Příklad – Malá násobilka

- Cykly můžeme vnořovat do sebe:

```
print("Malá násobilka pomocí cyklu:\n")
for i in range(1,11):
    for j in range(1,11):
        print(f"\t{i*j}",end=" ")
    print("")
```

Příklad

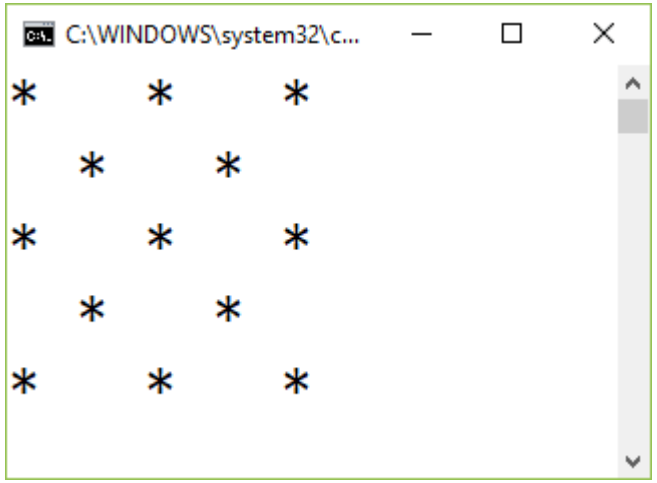
- Napíšeme program pro tisk čtverce o straně n složený z hvězdiček:

```
n = 5
for i in range(n):
    for j in range(n):
        print("*",end=" ")
    print("")
```

Příklad

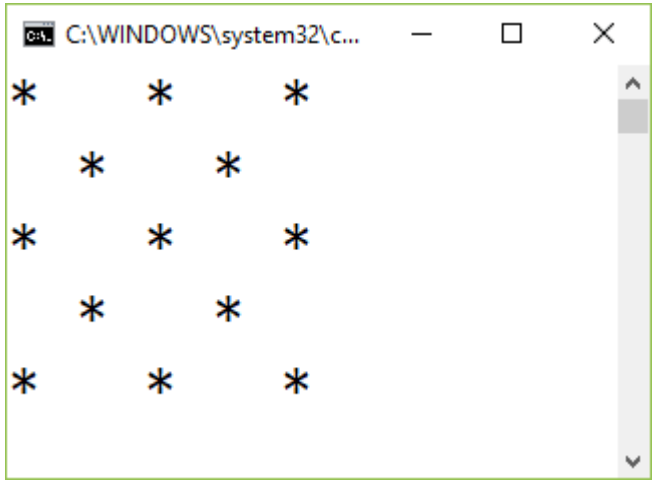
- Program modifikujeme pro tisk šachovnice z hvězdiček:

```
n = 5
for i in range(n):
    for j in range(n):
        if (i % 2 == 0 and j % 2 == 0):
        print("*",end=" ")
        elif (i % 2 == 1 and j % 2 == 1):
        print("*",end=" ")
        else:
        print(" ",end=" ")
    print("")
```



Příklad

```
n = 5
for i in range(n):
    for j in range(n):
        if ((i % 2 == 0 and j % 2 == 0) or (i % 2 == 1 and j % 2 == 1)):
        print("*",end=" ")
        else:
        print(" ",end=" ")
    print("")
```



Úkol

- Prvočíselné dvojče je prvočíslo, které je bud o dva vetší, nebo o dva menší než jiné prvočíslo. Vytisknete každou dvojici těchto prvočísel, které jsou menší nebo rovno 100.
- **Příklad výstupu:**

```
3 5
5 7
11 13
17 19
29 31
41 43
59 61
71 73
```