Řídící struktury

PVA2 Programování a vývoj aplikací

Adam Fišer

Obsah

Úvod

- Binární operace
- Podmínky
- Cykly
  - for
  - while
  - do while

Porovnávací operace

- `<` menší než
- `>` větší než
- `==` rovná se
- `>=` větší nebo rovno
- `<=` menší nebo rovno
- `!=` nerovnají se

Logické operátory

- `and` logický součin - obě hodnoty musejí být splněny
- `or` logický součet - alespoň jedna hodnota musí být splněna
- `not` negace hodnoty

Členské operátory

- `in` - je prvek v kolekci
- `not in` - není prvek v kolekci

Podmínky

Podmínky `if`

- Podmínka je výrok, který je vyhodnocen na pravda nebo nepravda
- Příkazy jsou vykonány pouze pokud je podmínka splněna
- Příkazy musí být odsazeny
- `if` je klíčové slovo
- `:` ukončuje podmínku

```
1   if (podminka):
2       # Při splnění podmínky je vykonán program za dvojtečkou
3       # Příkazy musí být odsazeny
```

Podmínky `if` - příklad

```
1   a = 200
2   b = 100
3   if (a > b):
4       print("a je větší než b") # Vypíše "a je větší než b"
```

`else`

- `else` - vykoná se pokud podmínka není splněna

```
a = 200
b = 100
if b > a:
  print("b je vetsi nez a")
else:
  print("a je vetsi nez b")
```

elif

- `else` - vykoná se pokud podmínka není splněna

```
a = 200
b = 100
if b > a:
  print("b je vetsi nez a")
elif a == b:
  print("a a b jsou stejne")
else:
  print("a je vetsi nez b")
```

Skládání podmínek - `and`

- `and` - logický součin
- Výsledek je `True` pouze pokud jsou obě podmínky `True`

```
a = 200
b = 100
c = 300

if (a > b and c > a):
  print("Obě podmínky jsou splněny")
```

```
a= 1
b= 0

if( a==1 and b==1 ): # Vypíše ne
    print("ano")
else:
    print("ne")
```

Skládání podmínek - `or`

- `or` - logický součet
- Výsledek je `True` pokud je alespoň jedna podmínka `True`

```
1   if( a==1 or b==0 ): # Vypíše ano
2       print("ano")
3   else:
4       print("ne")
```


Cykly

Cyklus

- Cyklus je opakování bloku kódu
  - `for` - opakuje blok kódu určitý počet krát
  - `while` - opakuje blok kódu dokud je podmínka splněna
  - `do while` - opakuje blok kódu dokud je podmínka splněna

Cyklus `for`

- Cyklus for se používá k procházení prvky určité posloupnosti, jako jsou seznamy, n-tice, řetězce nebo slovníky.

```
for i in posloupnost:
    # blok kódu je odsazen
    # provádějte kód s prvkem
    # tento kód se opakuje pro každý prvek v posloupnosti
```

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

Výstup:

```
0
1
2
3
4
```

Vsuvka - funkce `range()`

- `range()` - vytvoří posloupnost čísel

```
1  # funkce range generuje sekvenci celých čísel
2  range(start, stop, step)
3  Stop je povinný
4
5  # cyklus
6  # Funkce range(5) vrací seznam[0,1,2,3,4]
7  # Pro každé číslo i v seznamu
8  for i in range(5):
9    print(i)
```

Cyklus `for` - příklad

- Zobrazení hodnot seznamu

```
1   seznam = [10, 50, 75, 80]
2
3   # Pro každý prvek v seznamu
4   # Musíme nejprve zjistit délku seznamu len(seznam) a poté prvky projdeme
5   for x in range( len(seznam) ):
6       print(seznam[x])
7
8   # Cyklus vrátí:
9   10
10  50
11  75
12  80
```

Cyklus `for` - příklad přes text

- Zobrazení znaků řetězce

```
1  # Python texty jsou hodně podobné seznamům.
2  hello_world = "Hello, World!"
3
4  for ch in hello_world:    # Vytiskne každý znak z hello_world
5      print(ch)
```

Cyklus `for` - příklad pro kolekce

- V python sice nemáme klasický for each, ale můžeme použít klasický for pro projití kolekce

Seznam

```
1   fruits = ['jablko', 'banán', 'třešeň']
2   for fruit in fruits:
3       print(fruit)
```

Výstup:

```
1   jablko
2   banán
3   třešeň
```

Slovník

```
1   person = {'name': 'Adam', 'age': 40, 'salary': 13500.0}
2   for key, value in person.items():
3       print(key, value)
```

Výstup:

```
1   name Adam
2   age 40
3   salary 13500.0
```

Možnosti řízení toku

- `break`
- `continue`
- `pass`
- `else`

`break`

- `break` - ukončí cyklus a pokračuje na další kód

```
for i in range(5):
    if i == 3:
        break
    print(i)
```

Výstup:

```
0
1
2
```

`continue`

- `continue` - přeskočí zbytek kódu v cyklu a pokračuje další iterací

```
for i in range(5):
    if i == 3:
        continue
    print(i)
```

Výstup:

```
0
1
2
4
```

pass

- `pass` - pouze přeskočí blok kódu
- často se používá na místě, kde chcete mít správnou syntaxi a definujete strukturu kódu, ale ještě nechcete psát kód.

```
for i in range(5):
    if i == 3:
        pass
    print(i)
```

Výstup:

```
0
1
2
3
4
```

`else`

- `else` - blok kódu, který se vykoná po dokončení cyklu

```
for i in range(5):
    print(i)
else:
    print("Konec")
```

Výstup:

```
0
1
2
3
4
Konec
```

Cyklus `while`

- Cyklus while opakovaně provádí blok kódu, dokud je splněná určitá podmínka.
- Nutné ošetřit aby podmínka byla někdy splněna – riziko nekonečné smyčky.

```
while podminka:
    # vykonávaný kód cyklu je odsazen
    # tento kód se opakuje, dokud je splněna podmínka
    # podmínka se kontroluje na začátku každé iterace
```

```
sum = 1
while sum ⩽ 10:
    print(sum) # postupně vypíše narůstající součet 1,2,3,...10
    sum += 1

print("Konec")
```

Cyklus do while

- Cyklus do while je podobný cyklu while, ale podmínka se kontroluje až po prvním provedení bloku kódu.
- Používá, když chcete, aby se blok kódu vykonal alespoň jednou, i když podmínka není splněna.

```
while True:
    print("Hello, World!")
    break
```

Výstup:

```
Hello, World!
```

Děkuji za pozornost

Otázky?

Repository / Prezentace