Podprogramy - funkce a procedury

PLIN048 – Základy programování pro humanitní obory

Richard Holaj
FF MU

22. března 2017

Co nás dnes čeká?

- Princip podprogramů
  - Podprogramy a izolované bloky
  - Volání podprogramu
- Vstup a výstup podprogramu
  - Parametry a jejich paměťová reprezentace
  - Princip návratových hodnot
  - Funkce a typování
- Cvičení

Podprogramy a izolované bloky

- DRY princip
- izolované bloky
- lze pracovat jako s proměnnými (closures)

Definice podprogramu

```
1 def hello_everyone():
2     print("Hello world!")
3     print("Hello universe!")
4     print("Hello everyone!")
5 hello_world = lambda: print("Hello world!")
```

```
1 function hello_everyone() {
2     console.log("Hello world!");
3     console.log("Hello universe!");
4     console.log("Hello everyone!");
5 }
6 hello_world = function(){console.log("Hello world!")}
```

Volání podprogramu

Volání funkce

```
1 hello_everyone()
2 hello_again = hello_everyone
3 hello_again()
4 hello_everyone()
```

Volání funkce (izolovanost)

```
1 def test():
2     a = 10
3     print(a)
4
5 a = 20
6 test()
7 print(a)
```

Parametry a jejich paměťová reprezentace

```
1  def test(a, b, operation="+"):
2      if operation == "+" :
3          print(a + b)
4      elif operation == "-":
5          print(a - b)
6      elif operation == "*":
7          print(a * b)
8      elif operation == "/":
9          print(a / b)
10     else:
11         print("Unknown operation")
12
13 test(10, 20)
14 test(20, 30)
15 test(10, 20, "-")
16 test(10, 20, "*")
17 test(10, 20, "a")
```

Interpretace parametrů

Kód

```
1  def test(x):
2      print(x[0])
3      x[1] = 10
4      print(x[1])
5      x = [2, 4]
6      x[0] = 5
7      print(x[0])
8      print(x[1])
9
10 y = [1, 3]
11 print(y[0])
12 test(y)
13 print(y[0])
14 print(y[1])
```

Interpretace

```
1  y = [1, 3]
2  print(y[0])
3  x = y
4  print(x[0])
5  x[1] = 10
6  print(x[1])
7  x = [2, 4]
8  x[0] = 5
9  print(x[0])
10 print(x[1])
11 print(y[0])
12 print(y[1])
```

Princip návratových hodnot

- výstup
- rozlišení funkce x procedura
- *return*
- podobně jako v matematice
- funkce $\rightarrow$ výstup

Funkce s návratovou hodnotou

```
1 def odd_or_even(number):
2     if number % 2 == 0:
3         return "odd"
4     else:
5         return "even"
6
7 print("Number 10 is " + odd_or_even(10) + ".")
8 print("Number 9 is " + odd_or_even(9) + ".")
```

Interpretace funkce s návratovou hodnotou

```
1  number = 10
2  if number % 2 == 0:
3      odd_or_even = "odd"
4  else:
5      odd_or_even = "even"
6  print("Number 10 is " + odd_or_even + ".")
7  number = 9
8  if number % 2 == 0:
9      odd_or_even = "odd"
10 else:
11     odd_or_even = "even"
12 print("Number 9 is " + odd_or_even + ".")
```

Funkce a typování

```
1 String odd_or_even(int number) {
2     if(number % 2 == 0) {
3         return "odd"
4     } else {
5         return "even"
6     }
7 }
8 println("Number 10 is " + odd_or_even(10) + ".")
9 println("Number 9 is " + odd_or_even(9) + ".")
```

Funkce a typování – interpretace

```
1  int number = 10
2  String odd_or_even = ""
3  if(number % 2 == 0) {
4      odd_or_even = "odd"
5  } else {
6      odd_or_even = "even"
7  }
8  println("Number 10 is " + odd_or_even + ".")
9  int number = 9
10 if(number % 2 == 0) {
11     odd_or_even = "odd"
12 } else {
13     odd_or_even = "even"
14 }
15 println("Number 9 is " + odd_or_even + ".")
```

Cvičení

Mějme funkci *play* s parametrem *tracks* obsahujícím název skladeb v playlistu, parametrem *current* obsahujícím index aktuální skladby a parametrem *repeat*, který říká, zda chceme po přehrání poslední skladby pokračovat znovu od začátku. Implementujte funkci tak, že bude tento playlist „přehrávat“, kdy přehráním jedné skladby se rozumí výpis jejího názvu.

Implementujte funkce *prev* a *next* a druhou jmenovanou použijte v rámci funkce *play*.

Cvičení

Implementujte jednoduchý EventHandler tj. funkci *listen* **s parametry** *eventName* **typu text a** *listener* **typu funkce (s funkcemi lze pracovat jako s proměnnými) a funkci** *fireEvent* **s parametry** *eventName* **typu text a** *data* **typu slovník. Pro uložení listenerů použijte slovník klíčovaný podle jména události mající jako hodnotu seznam přiřazených funkcí. Funkce** *listen* **přidává funkci do seznamu podle klíče** *eventName* **a funkce** *fireEvent* **spustí postupně všechny funkce uložené pod tímto klíčem.**