

Εργαστήριο 4

Εισαγωγή στον Προγραμματισμό Υπολογιστών

Περιεχόμενα

1. Συναρτήσεις ως ορίσμα συνάρτησης
 - Γενικός υπολογισμός με τους αριθμούς 2, 3
 - Δέντρα από *
2. Συναρτήσεις ως τιμή συνάρτησης
 - Συνάρτηση που επιστρέφει ενσωματωμένη συνάρτηση
 - Συνάρτηση που επιστρέφει φωλιασμένη συνάρτηση
 - Currying
3. Συναρτήσεις στα ορίσματα και στην τιμή επιστροφής
 - Άθροισμα συναρτήσεων
 - Αντιστροφή ακολουθίας
4. Εκφράσεις λ

Γενικός υπολογισμός με τους αριθμούς 2, 3

- Συμπληρώστε τα κενά

```
>>> def apply_to_2_3(_____):
```

```
    _____
```

```
>>> apply_to_2_3(max)
```

```
3
```

```
>>> apply_to_2_3(pow)
```

```
8
```

```
>>> from operator import *
```

```
>>> apply_to_2_3(add)
```

```
5
```

Γενικός υπολογισμός με τους αριθμούς 2, 3

- Λύση

```
>>> def apply_to_2_3(func):  
    return func(2, 3)
```

```
>>> apply_to_2_3(max)
```

```
3
```

```
>>> apply_to_2_3(pow)
```

```
8
```

```
>>> from operator import *
```

```
>>> apply_to_2_3(add)
```

```
5
```

Δέντρα από *

- Φτιάξτε τη συνάρτηση `print_tree(n, level)` που εμφανίζει «δέντρα» με n επίπεδα όπου στο i -οστό επίπεδο εμφανίζεται `level(i)` πλήθος από *, πχ.

```
>>> def linear(i):  
    return i
```

```
>>> print_tree(3, linear)
```

```
*
```

```
**
```

```
***
```

```
>>> print_tree(6, linear)
```

```
*
```

```
**
```

```
***
```

```
****
```

```
*****
```

```
*****
```

```
>>> print_tree(3, square)
```

```
*
```

```
****
```

```
*****
```

```
>>> print_tree(6, square)
```

```
*
```

```
****
```

```
*****
```

```
*****
```

```
*****
```

```
*****
```

Δέντρα από *

- Λύση

```
>>> def print_tree(n, level):  
    i = 1  
    while i <= n:  
        print(level(i) * '*')  
        i = i + 1
```

Δέντρα από *

- Συμπληρώστε το κενό. Μπορείτε να ορίσετε βοηθητικές συναρτήσεις

>>>

*

* *

* * *

*

* *

* * *

*

* *

* * *

Δέντρα από *

- Λύση

```
>>> print_tree(9, periodic_3)
```

```
*
```

```
**
```

```
***
```

```
*
```

```
**
```

```
***
```

```
*
```

```
**
```

```
***
```

```
def periodic_3(x):  
    """1, 2, 3, 1, 2, 3, ... """  
    return (x + 2) % 3 + 1
```


Συναρτήσεις ως τιμή συνάρτησης

- Συμπληρώστε το κενό

```
>>> def foo():
```

```
>>> f = foo()
```

```
>>> f('hello')
```

```
hello
```

```
>>> f('hello world')
```

```
hello world
```

Συναρτήσεις ως τιμή συνάρτησης

- Λύση

```
>>> def foo():  
    return print
```

```
>>> f = foo()
```

```
>>> f('hello')
```

```
hello
```

```
>>> f('hello world')
```

```
hello world
```

Συναρτήσεις ως τιμή συνάρτησης

```
>>> def foo(n):
```

```
    _____  
    _____  
    _____  
    _____
```

```
>>> f = foo(2) # η foo επιστρέφει φωλιασμένη συνάρτηση
```

```
>>> f('hello') # ... η οποία λειτουργεί όπως η print
```

```
hello hello
```

```
>>> f('hello world')
```

```
hello world hello world
```

Συναρτήσεις ως τιμή συνάρτησης

```
>>> def foo(n):  
    def print_many(s):  
        print((n - 1) * (s + ' ') + s)  
    return print_many
```

```
>>> f = foo(2) # η foo επιστρέφει φωλιασμένη συνάρτηση  
>>> f('hello') # ... η οποία λειτουργεί όπως η print  
hello hello  
>>> f('hello world')  
hello world hello world
```

Currying

- Γράψτε συνάρτηση `times (n)` η οποία λειτουργεί ως εξής:

```
>>> def times (n, symbol):
```

```
_____
_____
_____
```

```
>>> times (3, '*')
```

```
'***'
```

```
>>> times (5, 'o')
```

```
'ooooo'
```

Currying

- Γράψτε συνάρτηση `times (n)` η οποία λειτουργεί ως εξής:

```
>>> def times (n, symbol):  
    return n * symbol
```

```
>>> times (3, '*')  
'***'
```

```
>>> times (5, 'o')  
'ooooo'
```

Currying

- Γράψτε τον μετασχηματισμό *Curry* της `times (n, symbol)`, δηλαδή συνάρτηση `curried_times (n)` η οποία λειτουργεί ως εξής:

```
>>> def curried_times (n):
```

```
    _____  
    _____  
    _____
```

```
>>> curried_times (3) ('*') # τι επιστρέφει η curried_times (3);  
'***'
```

```
>>> curried_times (5) ('o')  
'ooooo'
```

Currying

- Γράψτε τον μετασχηματισμό *Curry* της `times (n, symbol)`, δηλαδή συνάρτηση `curried_times (n)` η οποία λειτουργεί ως εξής:

```
>>> def curried_times(n):  
    def func(symbol):  
        return n * symbol  
    return func
```

```
>>> curried_times(3)('*') # τι επιστρέφει η curried_times(3);  
'***'  
>>> curried_times(5)('o')  
'ooooo'
```


Συναρτήσεις υψηλότερου επιπέδου

```
>>> def function_add(f, g):  
    """Άθροισμα συναρτήσεων."""  
    def ____ (x):  
        return _____  
    return ____  
  
>>> def square(x):  
    return x ** 2  
  
>>> def one_more(x):  
    return x+1  
  
>>> func = function_add(square, one_more) # x ** 2 + x + 1  
>>> func(2)
```

Συναρτήσεις υψηλότερου επιπέδου

```
>>> def function_add(f, g):  
    """Άθροισμα συναρτήσεων."""  
    def h(x):  
        return f(x) + g(x)  
    return h  
  
>>> def square(x):  
    return x*x  
>>> def one_more(x):  
    return x+1  
>>> func = function_add(square, one_more) # x ** 2 + x + 1  
>>> func(2)  
7
```

Εκφράσεις λ (lambda)

- Γράψτε ισοδύναμες εκφράσεις λ για τις επόμενες συναρτήσεις

```
"""def f(x):  
    return x ** 2 + 2 * x + 7
```

```
def add(f, g):  
    def h(x):  
        return f(x) + g(x)  
    return h
```

```
"""
```

```
f = lambda _____
```

```
add = _____
```

Εκφράσεις λ (lambda)

- Γράψτε ισοδύναμες εκφράσεις λ για τις επόμενες συναρτήσεις

```
"""def f(x):  
    return x ** 2 + 2 * x + 7
```

```
def add(f, g):  
    def h(x):  
        return f(x) + g(x)  
    return h
```

```
"""
```

```
f = lambda x: x ** 2 + 2 * x + 7
```

```
add = lambda f, g: lambda x: f(x) + g(x)
```

Συναρτήσεις υψηλότερου επιπέδου

- Υλοποιήστε συνάρτηση `reverse(term, n)` που επιστρέφει την *αντίστροφη ακολουθία* της `term`, δηλ. η `term(1), term(2), ..., term(n)` αντιστρέφεται σε `term(n), term(n-1), ..., term(1)`, πχ

```
>>> antilinear = reverse(linear, 5)
```

```
>>> i = 1
```

```
>>> while i <= 5:
```

```
    print(linear(i), antilinear(i))
```

```
    i += 1
```

```
1 5
```

```
2 4
```

```
3 3
```

```
4 2
```

```
5 1
```

Συναρτήσεις υψηλότερου επιπέδου

- Λύση

```
>>> def reverse(term, n):  
    def reversed_term(i):  
        return term(n - i + 1)  
    return reversed_term
```

Συναρτήσεις υψηλότερου επιπέδου

- Συμπληρώστε το κενό

```
>>>
```

```
*****
```

```
****
```

```
***
```

```
**
```

```
*
```

Συναρτήσεις υψηλότερου επιπέδου

- Λύση

```
>>> print_tree(5, reverse(linear, 5))
```

```
*****
```

```
****
```

```
***
```

```
**
```

```
*
```