Laboratory 6: Higher order functions

Prerequisits: +passing functions to functions, default parameters

Contents

PEP8

To monitor the coding style we turn on the PEP8 style guide monitoring in Spyder by:

1. Going to preferences in Spyder menu
2. Clicking on `Editor` in the selection list on the left
3. Clicking on `Code Introspection/Analysis` (top right corner)
4. Ticking the box `Real-time code style analysis`
5. Clicking `Apply` and `OK` (bottom right corner)

Exercises

Create a file lab6.py and populate with the following functions:

1. A function `positive_places(f, xs)` that takes as arguments some function `f` and a list of numbers `xs` and returns a list of those-and-only-those elements `x` of `xs` for which `f(x)` is strictly greater than zero.

   Example 1:

   ```
   In [ ]: def my_f(x):
      ...:     return x ** 3
      ...:

   In [ ]: positive_places(my_f, [1, 2, -1, -2, 3, 42, -9])
   Out[ ]: [1, 2, 3, 42]

   In [ ]: positive_places(my_f, [1, 2, 3, 4, 5])
   Out[ ]: [1, 2, 3, 4, 5]

   In [ ]: positive_places(my_f, [])
   Out[ ]: []

   In [ ]: positive_places(my_f, [-1, -2, -3, -4, -5])
   Out[ ]: []
   ```

   Example 2:

   ```
   In [ ]: def f(x):
      ...:     return 2 * x + 4
      ...:

   In [ ]: positive_places(f, [10, 1, -3, -1.5, 0, 0.5])
   Out[ ]: [10, 1, -1.5, 0, 0.5]
   ```

2. Write a function `eval_f_0123(f)` that evaluates the function `f=f(x)` at positions `x=0`, `x=1`, `x=2` and `x=3`. The function should return the list `[f(0), f(1), f(2), f(3)]`.

   Example 1:

   ```
   In [ ]: def square(x):
      ...:     return x * x
      ...:
   ```

```
In [ ]: eval_f_0123(square)
Out[ ]: [0, 1, 4, 9]
```

Example 2:

```
In [ ]: def cubic(x):
   ...:     return x ** 3
   ...:

In [ ]: eval_f_0123(cubic)
Out[ ]: [0, 1, 8, 27]
```

Example 3:

```
In [ ]: def stars(x):
   ...:     return "*" * x
   ...:

In [ ]: eval_f_0123(stars)
Out[ ]: ['', '*', '**', '***']
```

3. A function `eval_f(f, xs)` which takes a function `f = f(x)` and a list `xs` of values that should be used as arguments for `f`. The function `eval_f` should apply the function `f` subsequently to every value `x` in `xs`, and return a list `fs` of function values. I.e. for an input argument `xs=[x0, x1, x2,..., xn]` the function `eval_f(f, xs)` should return `[f(x0), f(x1), f(x2), ..., f(xn)]`.

   Example 1:

```
In [ ]: def square(x):
   ...:     return x * x
   ...:

In [ ]: eval_f(square, [-1, 10, 20, 42])
Out[ ]: [1, 100, 400, 1764]
```

   Example 2:

```
In [ ]: import math

In [ ]: eval_f(math.sqrt, [1, 2, 4, 9])
Out[ ]: [1.0, 1.4142135623730951, 2.0, 3.0]
```

   Example 3:

```
In [ ]: def sign(x):
    ...:     if x > 0:
    ...:         return 1
    ...:     elif x < 0:
    ...:         return -1
    ...:     else:
    ...:         return 0
    ...:

In [ ]: sign(-1.1)
Out[ ]: -1

In [ ]: sign(0.1)
Out[ ]: 1

In [ ]: sign(0.0)
Out[ ]: 0

In [ ]: eval_f(sign, [-0.2, -0.1, 0, 0.1, 0.2, 0.3])
Out[ ]: [-1, -1, 0, 1, 1, 1]
```

4. A function `sum_f(f, xs)` that returns the sum of the function values of `f` evaluated at values `x0, x1, x2, ..., xn` where `xs=[x0,x1,x2,...,xn]`.

Example 1:

```
In [ ]: def f(x):
   ...:     return x
   ...:

In [ ]: sum_f(f, [1, 2, 3, 10])
Out[ ]: 16
```

Example 2:

```
In [ ]: def square(x):
   ...:     return x * x
   ...:

In [ ]: sum_f(square, [1, 2, 3, 10])
Out[ ]: 114
```

5. A function `box_volume_UPS(a,b,c)` that returns the volume of a box with edge lengths `a`, `b` and `c`. Inputs should be provided in inch, and the output should be expressed in inch^3.

   The standard dimensions of the UPS express box (small) are `a=13` inch, `b=11` inch and `c=2` inch.

   Your function should use these values for `a`, `b` and `c` unless others are provided.

   Examples:

```
In [ ]: box_volume_UPS()
Out[ ]: 286

In [ ]: box_volume_UPS(a=10, b=10, c=10)
Out[ ]: 1000

In [ ]: box_volume_UPS(c=5)
Out[ ]: 715
```

Then submit `lab6.py` by email with the subject lab 6 for automatic testing of this laboratory session.

Last updated: 2019-01-06 at 17:34

Return to Top