INDEX PAGE
Pages
Lab 1
FAQ
Lab 2
Lab 3
Lab 7
Lab 6
Testing tips
Lab 4
Lab 10
Lab 5
Module overview
Support
Installing and finding Python
Snippets
Lecture slides
Lab 8
Lab 9
First steps
Testing demo

Home | **Lab 8**

Laboratory 8: Various applications, plotting

Prerequisites: pylab/matplotlib: plot, legend, axis-labelling, writing png/pdf files, root finding

Contents

- PEP8
- Exercises

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 `lab8.py` with the following functions:

1. Write a function `f1(x)` which accepts an number `x` as input and computes and returns

   $$f_1(x) = \cos(2\pi x)\exp(-x^2)$$

2. Write a function `f2(x)` which accepts the number `x` as input and computes and returns

   $$f_2(x) = \log(x + 2.2)$$

   where `log` refers to the natural logarithm (the Python function name is `math.log`).

3. 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. This task was given already as a training exercise in lab 6 where we also provide some input and output examples.

   In this lab, we ask that you write the same function *without* usage of a `for` or `while` loop to practice list comprehension or the use of `filter`.

4. Write a function `create_plot_data(f, xmin, xmax, n)` which returns a tuple `(xs, ys)` where `xs` and `ys` are two sequences, each containing `n` numbers:

   $$\texttt{xs} = [x_0, x_1, \ldots, x_{n-1}] \quad \text{with} \quad x_i = x_{\min} + i\frac{x_{\max} - x_{\min}}{n-1}$$
   $$\texttt{ys} = [f(x_0), f(x_1), \ldots, f(x_{n-1})]$$

   The function is expected to work for any `n >= 2`.

   Examples (here the tuple of returned sequences is a tuple of lists):

```
In [ ]: def f(x):
...:     return x * 10
...:

In [ ]: create_plot_data(f, -1, 1, 2)
Out[ ]: ([-1.0, 1.0], [-10.0, 10.0])

In [ ]: create_plot_data(f, 0, 2, 5)
Out[ ]: ([0.0, 0.5, 1.0, 1.5, 2.0], [0.0, 5.0, 10.0, 15.0, 20.0])
```

```
In [ ]: def f(x):
...:     return 0
...:

In [ ]: create_plot_data(f, 0, 1.5, 4)
Out[ ]: ([0.0, 0.5, 1.0, 1.5], [0, 0, 0, 0])
```

5. Using pylab or matplotlib, write a function `myplot()` that computes f1(x) [by calling the function f1 of course, or -- even better -- calling `create_plot_data`] and plots f1(x) using 1001 points for x ranging from -2 to +2. The function should return `None`.

   Then extend this function `myplot` to also plot f2(x) in the same graph.

   To plot two or more curves on the same figure, just use the plot() command twice.

   Label x-axis and provide a legend showing the function name (i.e. `f1` and `f2`) for the two curves.

6. Extend the function `myplot()` so that it saves a png file of the graph (with name `plot.png`) and a pdf file of the graph (with name `plot.pdf`) through pylab/matplotlib commands.

   > (Note: Generally, it is better to use pdf files rather than png files as pdf files are based on vector graphics and produce higher print quality, and can be zoomed without appearing pixelated. On the other hand, png files are good choice, for example, to include in webpages.)

7. Use the pylab-navigation bar (at the bottom of the figure) to zoom into the image. For x > 0, what is the value `x` of the functions where f1(x) = f2(x)?

   > (If you are using the the IPython console and by default your plots appear `inline`, i.e. in the IPYthon console, then you cannot zoom into the figure. If so, you should use the command `%matplotlib qt` in the console. After you have done this, the next plot figure should appear in its own pop-up window, and allow you to zoom in and out in the figure.
   >
   > To switch back to figures showing in the IPython console, use `%matplotlib inline`).

   Using this graphical information, write a function `find_cross()` which uses `scipy.optimize.brentq` to find the value `x` (approximately) for which `f1(x) = cos(2 * pi * x) * exp(-x * x)` and `f2(x) = log(x + 2.2)` have the same value. We are only interested in the solution where `x > 0`.

   Your function `find_cross()` should return the approximation of the root that `scipy.optimize.brentq` returns (the default tolerance settings are okay).

8. Write a function `reverse_dic(d)` that takes a dictionary `d` as the input argument and returns a dictionary `r`. If the dictionary `d` has a key `k` and an associated value `v`, then the dictionary `r` should have a key `v` and a value `k`. (This is only expected to work for dictionaries that have a unique set of values although you do not need to check for this.)

Submit `lab8.py` using `lab8` as the subject line.

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

Return to Top

Page last modified Sat Dec 1 14:44:10 2018. |