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 7**

Laboratory 7: Dictionaries, Recursion, Rootfinding (Newton's method)

Prerequisites: +recursion, +dictionaries, +finite differences

---

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 `lab7.py` which provides the following functions:

1. A function `count_chars(s)` which takes a string `s` and returns a dictionary. The dictionary's keys are the set of characters that occur in string `s`. The value for each key is the number of times that this character occurs in the string `s`. Examples:

```
In [ ]: count_chars('x')
Out[ ]: {'x': 1}

In [ ]: count_chars('xxx')
Out[ ]: {'x': 3}

In [ ]: count_chars('xxxyz')
Out[ ]: {'x': 3, 'y': 1, 'z': 1}

In [ ]: count_chars('Hello World')
Out[ ]: {' ': 1, 'H': 1, 'W': 1, 'd': 1, 'e': 1, 'l': 3, 'o': 2, 'r': 1}
```

   Note that the order in which the key-value pairs are listed in the output dictionary is not important.

2. A function `derivative(f, x)` which computes a numerical approximation of the first derivative of the function `f(x)` using central differences. The value that the function returns is

$$\frac{f\left(x+\frac{\epsilon}{2}\right)-f\left(x-\frac{\epsilon}{2}\right)}{\epsilon} \quad \text{where} \quad \epsilon = 10^{-6}$$

   Example:

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

In [ ]: derivative(f, 0)
Out[ ]: 0.0

In [ ]: derivative(f, 1)
Out[ ]: 2.0000000000575113
```

```
In [ ]: derivative(f, 2)
Out[ ]: 4.000000000115023
```

Note that you may not get *exactly* the same numbers as shown above.

3. Modify the `derivative` function such that it takes an optional third parameters `eps` to represent the greek letter epsilon in the equation above. The parameter `eps` should default to 1e-6.

   Examples:

```
In [ ]: import math

In [ ]: derivative(math.exp, 0, eps=0.1)
Out[ ]: 1.000416718753101

In [ ]: derivative(math.exp, 0)
Out[ ]: 1.0000000000287557

In [ ]: derivative(math.exp, 0, eps=1e-6)
Out[ ]: 1.0000000000287557
```

4. A function `newton(f, x, feps, maxit)` which takes a function `f(x)` and an initial guess `x` for the root of the function `f(x)`, an allowed tolerance `feps` and the maximum number of iterations that are allowed `maxit`. The `newton` function should use the following Newton-Raphson algorithm:

```
while |f(x)| > feps, do
   x = x - f(x) / fprime(x)
```

   where `fprime(x)` is an approximation of the first derivative (df(x)/dx) at position `x`. You should use the `derivative` function develope above.

   If `maxit` or fewer iterations are necessary for `|f(x)|` to become smaller than `feps`, then the value for `x` should be returned:

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

In [ ]: newton(f, 1.0, 0.2, 15)
Out[ ]: 1.4166666666783148

In [ ]: newton(f, 1.0, 0.2, 15) - math.sqrt(2)
Out[ ]: 0.002453104305219611

In [ ]: newton(f, 1.0, 0.001, 15)
Out[ ]: 1.4142156862748523

In [ ]: newton(f, 1.0, 0.001, 15) - math.sqrt(2)
Out[ ]: 2.1239017571339502e-06

In [ ]: newton(f, 1.0, 0.000001, 15) - math.sqrt(2)
Out[ ]: 1.5949463971764999e-12
```

5. If more than `maxit` iterations are necessary for the function `newton`, then the `newton` function should raise the `RuntimeError` exception with the message: `Failed after X iterations` where X is to be replaced with the number of iterations:

```
In [23]: def g(x):
   ....:     return math.sin(x) + 1.1  # has no root!
   ....:

In [24]: newton(g, 1.0, 0.02, 15)
Traceback (most recent call last):

  File "<ipython-input-6-0a9db3f67256>", line 1, in <module>
    newton(g, 1.0, 0.02, 15)
```

```
  File "..lab7.py", line 16, in newton
    raise RuntimeError("Failed after %d iterations" % maxit)

RuntimeError: Failed after 15 iterations
```

The relevant line of Python to be executed if the number of `maxit` iterations is reached, is

```
raise RuntimeError("Failed after %d iterations" % maxit)
```

6. A function `is_palindrome(s)` which takes a string `s` and returns the value `True` if `s` is a palindrome, and returns `False` otherwise. (Note that the return value is *not* the string `"True"` but the special Python value `True` -- see the section True and False in the appendix below. The same applies to `False`.)

   A palindrome is a word that reads the same backwards as forwards, such as *madam*, *kayak*, *radar* and *rotator*.

   Hints for a suggested algorithm:

   - if `s` is an empty string, then it is a palindrome.
   - if `s` is a string with one character, then it is a palindrome.
   - if the first letter of `s` is the same as the last letter of `s`, then `s` is a palindrome if the remaining letters of `s` (i.e. starting from the second letter, excluding the last letter) are a palindrome.

   Examples:

   ```
   In [ ]: is_palindrome('rotator')
   Out[ ]: True

   In [ ]: is_palindrome('radiator')
   Out[ ]: False

   In [ ]: is_palindrome('ABBA')
   Out[ ]: True
   ```

   We treat small letters (e.g. `a`) and capital letters (e.g. `A`) as different letters for this exercise: the string `ABba` is thus *not* a palindrome.

   Suggestion: if you struggle with the concept of recursion, take some time to study the output of this recursive factorial computation.

Then submit `lab7.py` by email with the subject lab 7 for automatic testing of this laboratory session.

Appendix

True and False

`True` and `False` are special boolean values (of Python type `bool`), and different from strings. Here is some demonstration of this:

```
In [ ]: a = True

In [ ]: b = "True"

In [ ]: type(a)
Out[ ]: bool

In [ ]: type(b)
Out[ ]: str
```

In the function `is_palindrome()` above, you must return the bool value `True` or `False`, but not the string `"True"` or the string `"False"`.

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

Return to Top

Page last modified Sun Nov 25 22:45:20 2018. |