Winfried Auzinger
Kevin Sturm

Sommersemester 2019
15.05.2019

Übungsaufgaben zur VU Computermathematik

Serie 7

Test your code with examples.

Exercise 7.1: ***Strings.***

(a) Write a function which gets as an input three strings `s_in`, `s_find` and `s_replace`. The return value of the function is `s_out` where the string `s_out` consists of all words of `s_in` in which all occurring words `s_find` are replaced by `s_replace`.

(b) Write a function `f(s)` which takes a string s and removes all its spaces. Let the function have an optimal argument which if set to `True` capitalises the input string.

Exercise 7.2: ***Classes. I.***

- Write a class `Complex` with methods `add`, `multiply` and `divide` which realises the addition, multiplication and division of two complex numbers `z1` and `z2`. Define complex numbers by its real and imaginary parts and use python tuple, i.e., `z = (imag, compl)`. (Do not use the build in complex numbers of python). The constructor `__init__` should initialise `z1` and `z2`.

Exercise 7.3: ***Classes. II.***

(a) Write a class `Vector` with methods `add(z1, z2)` and `scalar(a, z1)` which realise the addition and scalar multiplication of two lists `z1` and `z2` and the scalar `a` and the vector `z1`, respectively.

(b) Write an inherited class of `Vector` named `VectorPlus` which additionally has the functions `vector_prod(z1, z2)` and `tensor(z1, z2)` realising the tensor and vector product of two lists `z1` and `z2`.

Exercise 7.4: ***Classes. III.***

The faculty `n!` can be approximated using the Stirling formula $\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$; ($e$ is the Euler number). A second way to approximate the faculty is by the formula

$$(z+1)! \approx \sqrt{\frac{2\pi}{z}}\left(\frac{1}{e}\left(z+\frac{1}{12z-\frac{1}{10z}}\right)\right)^z \tag{1}$$

(a) Write a class `faculty` which has the methods `fac(n)`, `fac_stir(n)`, `fac_gam(n)`, `fac_stir_err(n)`, `fac_gam_err(n)`. The function `fac` returns the exact faculty of the number n and `fac_stir(n)` and `fac_gam(n)` an approximation using Stiring's formal and (1), respectively. The methods `fac_stir_err` and `fac_gam_err` return the error the approximations `fac_stir` and `fac_gam`, respectively.

(b) Equip all methods of the class with an additional positional argument `ptr` (that means for instance `fac_stir(n,ptr=True)`), which is by default `False`. If `ptr` is set to `True`, then the result of the called method shall be printed with `print`.

Exercise 7.5: ***Decorators. I.***

(a) Write a decorator `dec(ev, fun)` which gets a function `fun` and returns a function that evaluates `fun` at `ev`. Test your code with the functions `gamma` and `exp` of the standard library `math`.

(b) Write a decorator `comp(fun1, fun2, fun3)` which returns the function decomposition of `fun1`, `fun2` and `fun3`.

Exercise 7.6: ***Decorators. II.***

- Let `f` be a python function. Write a decorator `count` which counts how often the function `f` was called. Test your program with `sin` and `cos` of the math library. Example: with `f = count(sin)` the call `f(0.1)` should return 1 and `sin(0.1)` and another call `f(0.2)` would return 2 and `sin(0.1)`. HINT: in the inner definition of the decorator define the 'counter' variable which counts the function calls as `nonlocal` (syntax: `nonlocal counter`). This makes the variable 'counter' available in the outer function definition.

Exercise 7.7: ***Doc String***

- Write a detailed doc string documentation for the classes Complex and Vector and their functions of the previous exercise. Test your code by call help in the console as well as calling the functions and the module with ".__doc__"!

Exercise 7.8: ***Exceptions***

(a) Read the Python online tutorial https://docs.python.org/3.6/tutorial/errors.html on exceptions. How are exception defined in python?

(b) Write a function division(x,y) which return x/y. Write an exception when the absolute value of y is smaller than 1e-14.