Erratalist for 4th Edition of A Primer on Scientific Programming with Python

Hans Petter Langtangen

Nov 18, 2015

Page 4. The one-line program to be written in a text editor must have t**2 and not t~2.

Page 8. ... $v_0$ in mathematics becomes v0 in the program. (Not v_0.)

Page 11. \%0xd is not an integer padded with x leading zeros, but an integer written with a minimum field width of x, where any leading spaces are replaced by zeros.

Page 81. The printout does not correspond to the show code. The correct printout is

Jan: 56.6
Feb: 72.7
Mar: 116.5
Apr: 153.2
May: 191.1
Jun: 198.5
Jul: 193.8
Aug: 184.3
Sep: 138.3
Oct: 104.6
Nov: 67.4
Dec: 52.4

The code can also be simplified

```python
monthly_mean = []
n = len(data) # no of years
for m in range(12): # counter for month indices
    s = 0 # sum
    for y in data: # loop over "rows" (first index) in data
        s += y[m] # add value for month m
    monthly_mean.append(s/n)
```
Page 95. In the program c2f.py, the `print` statement has a wrong formatting of the $F(C)$ value: `%.51f` must read `%5.1f`, i.e., the same formatting as used for the $C$ value.

Page 99. Heading in Section 3.1.5 should be `Function argument or global variable?` (or instead of `of`).

Page 130. In Exercise 3.6, Equation (3.10), the sum must be $\sum_{i=0}^{n-1}$.

Page 158. The terminal output for the integral $\int_{0}^{\pi/2} \sin x \, dx$ should read 1, not 0.583009.

Page 189. In the mymod.py module, an `import sys` is needed in the test block before `print add1(float(sys.argv[1]))`.

Page 189. The last code showing the test block of the `interest` module needs two corrections: division by 365 if the expression for `years` and an `f` in the `print` statement:

```python
if __name__ == '__main__':
    import sys
    p = float(sys.argv[1])
    years = days(1, 2, p)/365.0
    print 'With p=%.2f it takes %.f1 years to double' % (p, years)
```

Page 191, Section 4.9.5. The computational example in this section involves the parameters $A0=2$ and $A=1$, but it should be the other way around: $A0=1$ and $A=2$ (otherwise the interest rate becomes negative).

Page 197.

```
export PYTH=$HOME/software/lib/pymodules:$PATH
```

should be

```
export PATH=$HOME/software/lib/pymodules:$PATH
```

Page 216. In Exercise 4.17, the reference to the program `user_formula.py` should be `integrate.py`. The name of the resulting program is then better named `integrate2.py` than `user_formula2.py`.

Page 273. In the first code block, $x3 = \text{mat}(x).\text{transpose}()$ should be $x3 = \text{mat}(x1).\text{transpose}()$.

Page 315. End of first paragraph: `Sun` is to be replaced by `Apple`. 
Page 331. Last code block,

```python
station = line.split('</strong>')[0].split('<strong>')[0]
```

must be

```python
station = line.split('</strong>')[0].split('<strong>')[1]
```

Page 426. The argument in the `raise ValueError` call, after `if c*d <= 0`, needs a final `% other`.

Page 447. The title of the chapter should be *Random numbers and simple games*.

Page 508. Exercise 8.13: Since `4 ≤ n ≤ 10` balls are drawn, one must investigate cases where `n ∈ [4, 10]`, say `n = 4, 7, 10` (and not `n = 1, 5, 10, 20` which does not make sense).

Page 676. The last line of the `integrate_ode.py` program should not contain `u'(t)=t**3`, but read

```python
print "Numerical solution of u'(t)=%s: %.4f" % 
    (f_formula, integrate(T, n, u0))
```

The four terminal output sessions below are then also wrong: instead of `u'(t)=t**3` it should be `u'(t)=t**exp(t**2)`.

Page 756. Exercise E.1: The exact solution is `u(t) = 0.2e^{0.1t}`.

Exercise E.24: The first constant on the right-hand sides of equations (E.70) and (E.72) must be 3 and 4, not 2 and 3.

Page 775. Below the equation with `I(t + Δt)`, it must read `Δt → 0` (not `∞`).

Page 776. Exercise E.42: The last element of the returned list in the `ProblemSIR.__call__` method should not have a minus sign; it should be `self.nu(t)*I`.

Page 780. Exercise E.46: A value for `I(0)` is not given. Set `I(0) = 0`.

Page 836. The text says “The import statements can actually be dropped since functions from `numpy` and `matplotlib` are imported by default when running the notebook in the browser or by supplying the command-line argument `--pylab` when starting notebooks locally on your machine.”. Now, the use of `--pylab` is discouraged. Also, the functions from `numpy` and `matplotlib` are not any longer automatically imported - you have to do that explicitly.

The recommended way of using IPython notebooks with `numpy` and `matplotlib` is to do
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

If you want the notebook to behave more as Matlab and not use the `np` and `plt`
interface, you can instead write

`%pylab`

**Page 843.** To successfully execute the `c2f.py` program, `cmd` must be `python`
`c2f.py 21` or `./c2f.py 21` (if `c2f.py` is an executable file) unless `.` is in the
user's `PATH` variable.

**Page 844.** Under **Split file or folder name**, the directory name `user` in the
text should be replaced by `/home/hpl` according to the interactive session.