Programming Techniques II
Exercise 12

Task 12.1 Preparation: Python Modules

Modularising code will be useful when writing simulations in Python. With the following directory structure:

```
src/
    main.py
    hello.py
    include/
            world.py
```

You can include your own Python libraries via relative paths (assuming a class named *World* exists in `world.py`) just like the system-provided modules in `main.py`:

```
import hello
from include.world import World
```

Note: older Python versions require *namespace packages* to have an empty file named `__init__.py` in and leading up to any directory you wish to include modules from.
You can read a more detailed description of modules in the official documentation:
https://docs.python.org/3/tutorial/modules.html

Task 12.2 PyPenna

Write a simulation of the Penna model in Python.
Create an extensible directory structure parallel to your C++ implementation and write the simulation into `main.py`. Design and import a Sheep class from `zoo/sheep.py`.
The code has to produce results similar to the week 1 C++ exercise, but the output doesn't have to be identical (due to differences in the RNG libraries, which will be remedied next week).

Task 12.3 Python Golf Challenge (optional)

We are golfing. The task is to replicate the following output exactly.

- First, print the alphabet in capitalised and non-capitalised form adjacently:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
```

- Next, your program should print the letters interleaved:

```
AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz
```

- And again, but wrapped at every 10 characters, in a box (with spaces on both sides of each character):

```
__________________________
| A a B b C c D d E e |
| F f G g H h I i J j |
| K k L l M m N n O o |
| P p Q q R r S s T t |
| U u V v W w X x Y y |
| Z z                 |
__________________________
```

- Now we want to annotate the alphabet in a Python list at four specific points:

```
['a: ', 'apple']
['b']
['c']
['d']
['e']
['f']
['g: ', 'grape']
['h']
['i']
['j']
['k']
['l: ', 'lemon']
['m']
['n']
['o: ', 'olive']
['p']
['q']
['r']
['s']
['t']
['u']
['v']
['w']
['x']
['y']
['z']
```

- And finally we only want to see the relevant parts as a string:

```
apple begins with a, grape begins with g, lemon begins with l, olive begins with o
```

Write a correct Python 3 implementation in a single file **without** using the `import` statement or executing any external code.
The submission with the smallest size (in bytes) wins.

You can check it with the *wordcount* command line utility (`wc -c filename`). Note that you don't have to put everything on one line: most spaces and/or semicolons are interchangeable with newlines.

The submission deadline for this challenge is Wednesday 9.12.15 at 05:00 in the morning. Notify us via the mailing list (pt2_hs15_ta@lists.phys.ethz.ch) with a link to your uploaded solution on your PT2 repository on GitLab.