Exam Assignments 12

Extension Types in Python

Extension types are a way for programmers to wrap efficient C/C++ code in a way that makes it easy to use from python. They can be used like normal python objects, but can be much more efficient, as they are actually compiled code. In cython, they are defined using `cdef class <class_name>`.

Data Fields in Cython Extension Types and Python Classes

Data fields in cython extension types have to be declared at the class level (unlike python classes, which allow the programmer to declare new fields dynamically in code), as they are saved in a `struct` and not in a `dict`.

Extension type data fields are by default only accessible from cython code, and have to be declared `readonly` or `public` to also be accessible from python.

Extension type field access in cython is faster than field access in python classes.

Extension types also allow other extension types as fields.

Extension types may contain methods declared using `def`, `cdef` or `cpdef`. Their accessibility from python code varies between these declarations.

Wrapping C/C++ Code in Cython

If one has some C or C++ code one wishes to wrap in cython so they can use it in python code, one can take the following steps:

Assume one has some library in the form of a header file `lib.h` and a code file `lib.cpp`, defining some function `int foo(int n)` that one wants to make usable from python.

Create a cython file called, for example, `foo_lib.pyx`. This file should contain the following:

- At the top, the neccessary compiler directives:
    - the language used (in our case `language = c++`)

    - the sources (in our case `sources = lib.cpp`)
    - compiler and linker options to be used (for example `extra_compile_args = -ffast-math`)
    - the cython language level (for example `cython: language_level = 3`)
- Below that, the functions from the `lib.h` file to be imported:

```
cdef extern from "lib.h":
    int foo(int n)
```

- Below that, the functions that are supposed to be calleable from python, which use the imported C/C++ functions:

```
def _foo(int n):
    return foo(n)
```

This file `foo_lib.pyx` can now be compiled (for example using `cythonize`) and imported from python like any other cython file.