Ways to bind C structs to python classes.

1. Creating the python wrapper of a generic C struct:

```
1.1 class ndpi_detection_module_struct(Structure):
        pass
1.2 class u6_addr(Union):
        pass
```

2. Adding fields to the structure, two ways:

2.1 Setting _fields_ in the constructor

```
class Example_struct(Structure):
    _fields_ = [ ("example",c_uint8), ("name",c_uint16)]
```

2.2 Or after creating the struct:

```
example_truct._fields_ = [("example",POINTER(example_struct))]
```

Notes:

1. It is not possible to declare a pointer to *Example_struct* with the first method.
2. The order of the fields specified in C must be kept in python
3. In C, the fields present in the struct can be changed a compile time by the use of ifndef. All the possibilities must be handled by creating multiple python *Structure*.

Wrapping C functions.

1. Declare restype:

```
<var_name>.<function_name>.restype = <c type> #of the return value
```

2. Declare the type of the argument

```
<var_name>.<function_name>.argtypes = [ <c arg0 type>, <c arg1 type>, ... , <c argN type>]
```

Notes:

1. if the return type is *char** it is preferable to declare the restype as *c_void_p* and then convert it using cast.
2. These steps are optional in case of primitive C types, but it's highly recommended.