L03_modules_and_classes

Introduction

This assignment will cover how import and utilize a library and focus on how to create and understand classes in Python.

Python Basics

Modules and Packages in Python

Qa Load and test the `libitmal` module

Try out the `libitmal` module from [GITMAL]. Load this module and run the function

```
from libitmal import utils as itmalutils
utils.TestAll()
```

from this module.

Qa implementation

```
In [1]: # Assignment Qa:
        from libitmal import utils as itmalutils
        itmalutils.TestAll()
```

```
TestPrintMatrix...(no regression testing)
X=[[   1.    2.]
   [   3. -100.]
   [   1.   -1.]]
X=[[ 1.  2.]
   ...
   [ 1. -1.]]
X=[[   1.
       2.    ]
   [   3.0001
    -100.    ]
   [   1.
      -1.    ]]
X=[[   1.    2.]
   [   3. -100.]
   [   1.   -1.]]
OK
TEST: OK
ALL OK
```

Qa results

In order to use files from the directory libitmal, the user has to "navigate the OS" there. This can be done with System Environment Variables in Windows. Alternatively, the user can make use of the "import sys" library and write the path destination.

Qb Create your own module, with some functions, and test it

Now create your own module, with some dummy functionality. Load it and run you dummy function in a Jupyter Notebook.

Keep this module at hand, when coding, and try to capture reusable python functions in it as you invent them!

Qb implementation

```
In [2]: # Assignment Qb:
        from libitmal import ahs_utils as ahs
        ahs.HelloWorld()
```

```
hello world this is insane function
```

Qb results

It is important to restart the kernel if the library has been modified, otherwise it may not override current section in Jupyter Notebook.

Qc How do you 'recompile' a module?

When changing the module code, Jupyter will keep running on the old module. How do you force the Jupyter notebook to re-load the module changes?

Qc results

Simple solution is to open Jupyter Notebook, then click on the "Kernel" tab and then click on "Restart & Run All". This will restart the kernel and run all sections from top to bottom.

Classes in Python

Qe Extend the class with some public and private functions and member variables

How are private function and member variables represented in python classes?

What is the meaning of `self` in python classes?

What happens to a function inside a class if you forget `self` in the parameter list, like `def myfun():` instead of `def myfun(self):` ?

Qe implementation

In [3]:
```
# Assignment Qe
class MyClass:
    mPublic = "mPublic is: Gnomed" # Public

    def __init__(self, vProtected):
        self.mPublic = None
        self._mProtected = vProtected # Protected
        self.__mPrivate = None # Private

    def myfun(self):
        print("This is a message inside the class, printed from myfun(self).")

    def printProtectedSelf(self):
        print(self._mProtected)

    def NoSelf():
        return (self.__mPrivate, self._mProtected, self.mPublic)

    def Self(self):
        return (self.__mPrivate, self._mProtected, self.mPublic)

# Creating an object to the class and defining the argument vProtected in the __ini
t__ function
myobjectx = MyClass("YOU HAVE PRINTED A 'PROTECTED VARIABLE'")
myobjectx.myfun()

# How to print a Protected Variable
myobjectx.printProtectedSelf()
# Protected variable can be printed directly without an instance
print(myobjectx._mProtected)

# MyClass.printProtected() # Not allowed
# myobjectx.printProtected() # Not allowed: You need to create it with self in orde
r to make it work

# Print difference between Self(self) and NoSelf() functions in the class
print(myobjectx.Self())
# print(myobjectx.NoSelf()) # Not possible
```

```
This is a message inside the class, printed from myfun(self).
YOU HAVE PRINTED A 'PROTECTED VARIABLE'
YOU HAVE PRINTED A 'PROTECTED VARIABLE'
(None, "YOU HAVE PRINTED A 'PROTECTED VARIABLE'", None)
```

Qe results

Following the python script, it is possible to see which functions are allowed and not allowed. The use of the self-argument is also very important, otherwise it may lead to unusable functions with specific variables.

Qf Extend the class with a Constructor

Figure a way to declare/define a constructor (CTOR) in a python class. How is it done in python?

Is there a class destructor in python (DTOR)? Give a textual reason why/why-not python has a DTOR?

Qf implementation

In [4]:
```python
# Assignment Qf
class FooType():
    def __init__(self, id):
        self.id = id
        print(self.id, 'constructed')

    def __del__(self):
        print(self.id, 'destructed')


def CheckFooType():
    obj = FooType(1)
    obj2 = FooType(2)
    obj3 = FooType(3)

CheckFooType()
```

```
1 constructed
2 constructed
3 constructed
1 destructed
2 destructed
3 destructed
```

Qf results

The python code has shown how CTOR (`__init__`) and DTOR (`__del__`) are made in python. Furthermore, Destructors in python can be used to end some sort of connections, such as an internet connection.

Qg Extend the class with a to-string function

Then find a way to serialize a class, that is to make some `tostring()` functionality similar to a C++

```cpp
friend ostream& operator<<(ostream& s,const MyClass& x)
{
    return os << ..
}
```

Qg implementation

In [6]:
```python
# Assignment Qg:
class FooType():
    def __init__(self, id):
        self.id = id
        print(self.id, 'id constructed')

    def __del__(self):
        print(self.id, 'id destructed')

    def changeId(self, var):
        self.id = var

    def __str__(self):
        return str(self.id) + " id -- Class toString() in python"


p = FooType(1)
p.changeId(4)
print(p)
```

```
1 id constructed
4 id destructed
4 id -- Class toString() in python
```

Qg results

With the use of `_str__(self)` function in the class, we are able to serialize the class similar to toString() in C++.

Conclusion

Thus we were succesfully able to import and use the libraries in the directory "libitmal" by defining the System Environment Variable to the path in Windows.

Furthermore, we have explored that Classes in Python must be constructed with care and the use of pre-implemented functions such as `___init__`, `___del___`, `__str___` and so on.