Object Oriented Programming with Python

Gramsci Hermozo

Session 10

Content

- Inheritance
- Polymorphism

Abstract Classes

By default Python does not provide abstract classes. Python had/comes with a module that provides the base for defining Abstract Base classes.

```
# Firts thing that need to import to use abstract classes
from abc import ABC

class MyABC(ABC):
  pass
```

Inheritance

Multiple Inheritance

A class can be derived from more than one base class in python, similar to c++.

```
class Base1:
  pass

class Base2:
  pass

class MultiDerived(Base1, Base2):
  pass
```

Polymorphism

What it's?

Is the ability to perform an action on an object regardless of its type. This is generally implemented by creating base class and having two or more subclasses that all implement methods with the same signature.

```
class Shape(object):
  def calculate_area(self):
    raise NotImplemented

class Square(Shape):
  def calculate_area(self):
    pass

class Triangle(Shape):
  def calculate_area(self):
    pass
```

Polymorphism

Overloading

```
# functions that could be overload
__add__(self, other) # a1 + a2
__sub__(self, other) # a1 - a2
__mul__(self, other) # a1 * a2
# more operators
__int__(self) # int(a1)
```

Pomorphism

Overriding

```
class Parent(object):
  def say_hi(self):
    print("Hello")

  def introduce(self):
    print("I am the Parent")

class Child(Parent):
  def introduce(self):
    print("I am the Child")
```

Polymorphism

Lets practice

- Implement your static class (helper) that return lenght of an string, list and dictionary.
- Implement your own list structure that let the user:
  - Add element into it
  - Remove elements from it
  - Add support to use the helper and send this object to calculate the lenght