UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Arhitectura sistemelor de calcul

28. Descrierea notiunilor de clase de obiecte

Clase si obiecte

Trebuie subliniat ca in Python, cuvantul “obiect” nu se refera neaparat la instantierea unei clase. Clasele in sine sunt obiecte, iar, in sens mai larg, in Python toate tipurile de date sunt obiecte. Exista tipuri de date care nu sunt clase: numerele intregi, listele, fisierele. Toate tipurile de date insa au aproximativ acelasi comportament, mai usor de explicat daca ne referim la aceste tipuri de date folosind cuvantul “obiect”.

Un obiect se creeaza in Python prin folosirea cuvantului cheie `class`:

```
1. class className [(super_class1 [, super_class2]*)]:
2.     [interiorul clasei]
```

creaza un obiect de tip *clasa* si ii da numele *className*. *Interiorul clasei* poate contine definitii de metode locale si atribuiri pentru variabile locale. Clasa este derivata din *super_class1* si din *super_class2*.

```
1. class MyClass (object): ...
```

Creaza o clasa de tip “new-style” prin mostenire din object. Vechiul tip de clase nu mostenesc object.

In lucrul cu clase, trebuie stiute urmatoarele reguli:

- Primul argument pentru metodele (operatiile) de instanta ale unei clase este intotdeauna obiectul sursa, numit “self” prin conventie. Cand ne referim la membri ai clasei, vom folosi `self.membru`, intr-un mod asemanator cu folosirea “this” din Java.
- Metoda speciala *`init`*`()` este apelata la instantierea clasei (crearea unui obiect de tipul clasei) si poate fi asemuita cu un constructor.
- Metoda speciala *`del`*`()` este apelata cand nu mai sunt referinte la acest obiect (mecanism de garbage collector) si poate fi asemuita cu un destructor.
- Instantierea se face prin apelarea obiectului clasa, posibil cu argumente (astfel `instance=apply(aClassObject, args…)` creaza o instanta). Spre exemplu:

```
1. class Complex:
2.     def __init__(self, realpart, imagpart):
3.         self.r = realpart
4.         self.i = imagpart
5.
6. x = Complex(3.0, -4.5)
```

De multe ori vom implementa clase derivand din clasa `'Thread'`, caz in care ar trebui respectate urmatoarele reguli:

- Subclasati threading.Thread()
- Suprascrieti *init*() si run()
- Nu suprascrieti start()
- In *init*(), apelati Thread.*init*()