

OOP CURS 2

Silviu Ojog

LINK***Academy***

Recapitulare?

- Ce știm până acum despre Python?
 - Limbaj interpretat?

Recapitulare?

- Ce știm până acum despre Python?
 - Limbaj interpretat?
 - Limbaj obiect orientat? (Object-Oriented-Programming)

Recapitulare

- Ce știm până acum despre OOP?
 - Obiect vs clasa?
 - Metode vs functii?
 - Ce înseamnă self?
 - Ce înseamnă `__init__`?

Exercitiu

- Creati o clasa pentru categoria Masina care sa contina:
 - Marca, Consum, Nr de km parcursi, Pret combustibil
 - Alimentare benzina si combustibil disponibil
 - Parcurgere kilometrii
 - Bani cheltuiti cu masina (consumul)

Interactiunea intre clasei

Clasa Student

- varsta
- nr_telefon
- nume
- note

Numar (integer)

String

String

Array de numere

Interactiunea intre clasei

Clasa Student

- varsta
- nr_telefon
- nume
- note
- facultate(program de studiu)

Numar (integer)

String

String

Array de numere

?????

Interactiunea intre clasei

Clasa Student

- varsta
- nr_telefon
- nume
- note
- facultate(program de studiu)

Numar (integer)

String

String

Array de numere

Clasa Program

- nume
- cursuri
- profesori

emy

Interactiunea intre clasei

Clasa Student

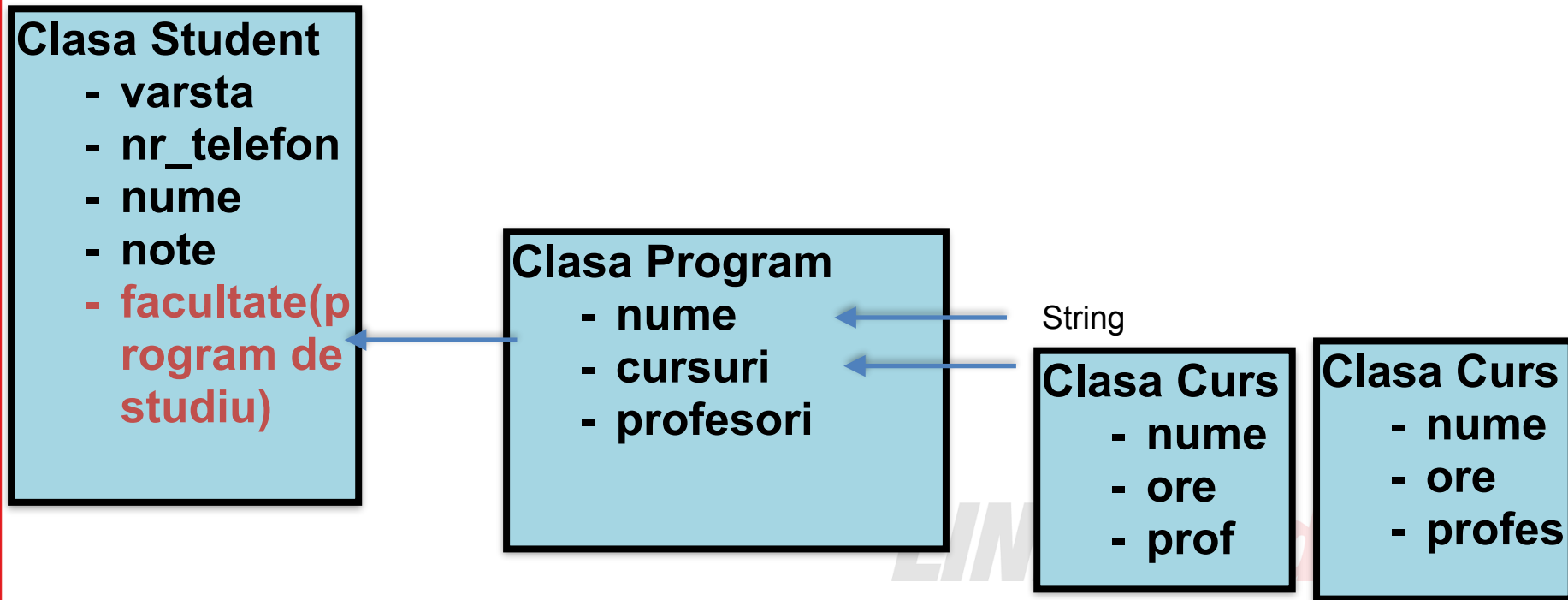
- varsta
- nr_telefon
- nume
- note
- facultate(program de studiu)

Clasa Program

- nume
- cursuri
- profesori

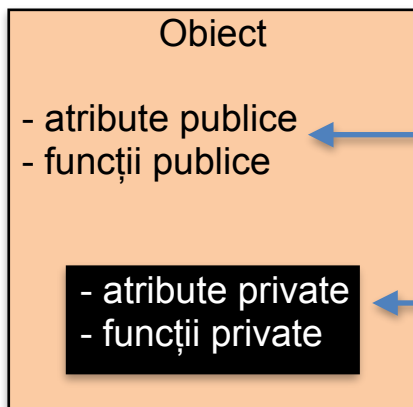
String

Interactiunea intre clasei



Încapsulare

Procesul prin care ținem datele și funcțiile separate de exterior. (atribute și funcții private)



Accessible oriunde, în cadrul definirii clasei, și în exteriorul ei

Accessible doar în cadrul definirii clasei

Componentele private

- By default în multe limbaje de programare metodele și atribuibile sunt private.
- Componentele private pot fi “accesate” din exteriorul obiectului cu ajutorul unor metode publice. (*getter* și *setter*)
- Scopul este de *verifica* valorile atribuite.

Componentele private

Clasa Student

- varsta
- nr_telefon
- nume
- note

Cel puțin 16/18 ani?

Maxim 120 ani?

Nu pot fi insert valori de tip "@#@"
"kjds", "-"

Componentele private

Clasa Student

- varsta
- nr_telefon
- nume
- note

Doar numere, cel puțin 10 caractere

Componentele private

Clasa Student

- varsta
- nr_telefon
- nume
- note

Doar caractere din alfabet
Cel puțin 3 caractere

Componentele private

Clasa GrupaStudent

- nume
- lista_studenti
- orar
- lista_profesori

Componentele private

Obiect GrupaStudent

- nume
- lista_studenti
- cursuri
- orar
- lista_profesori

Obiect Student

- varsta
- nr_telefon
- nume
- note

Obiect Student

- varsta
- nr_telefon
- nume
- note

Obiect Student

- varsta
- nr_telefon
- nume
- note

Componentele private

Obiect GrupaStudent

- nume
- lista_studenti
- cursuri
- orar
- lista_profesori

Obiect Student

- varsta
- nr_telefon
- nume
- note

Obiect Student

- varsta
- nr_telefon
- nume
- note

Obiect Student

- varsta
- nr_telefon
- nume
- note

Obiect Profesor

- nr_telefon
- nume
- email

Obiect Profesor

- nr_telefon
- nume
- email

Încapsulare

```
class Student:
```

```
    def setMarriageStatus(self, isMarried):
```

```
        self._isMarried = isMarried;
```

```
student1 = Student(...);
```

Metode (functii in clasa)

```
class Student:
```

```
    def setMarriageStatus(self, isMarried):
```

```
        self._isMarried = isMarried;
```

```
student1 = Student(...);
```

variabilă precedată de trebuie folosită doar în interiorul clasei



Attribute interne

Ex: `self._isMarried`

- In Python nu exista o modalitate de a face o variabila/metoda privata, spre deosebire de alte limbaje de programare
- O variabila privata are rolul de a nu fi schimbata in afara clasei in mod eronat (din greseala sau rau intentionat)

Convenții de nume

- Clasa (Class):
 - numele clasei incepe cu litera mare
- Metodele (functiile), Obiectele, Atributele:
 - numele lor incep cu litera mica
- Atribute interne
 - sunt precedate in denumire de _

Attribute interne

```
class Student:
```

```
    def setMarriageStatus(self, isMarried):
```

```
        self._isMarried = isMarried;
```

```
student1 = Student("Andrei", 45, 073762736276);
```

Verificarea atributului

```
class Student:
```

```
    def setMarriageStatus(self, isMarried):
```

```
        self._isMarried = isMarried;
```

```
    def getMarriageStatus(self):
```

```
        if hasattr (self, self._isMarried):
```

```
            return self._isMarried;
```

```
        else:
```

```
            return "Unknown";
```

```
student1 = Student("Andrei", 45, 073762736276);
```


Verificarea atributului

`hasattr (object, attribute):`

Funcția `hasattr` verifică dacă un anumit atribut se regăsește într-un anumit obiect

Varibila Secreta

```
class Student:
```

```
    def __init__(self, name, age, telephone):
```

```
        self.name = name;
```

```
        self.age = age;
```

```
        self.telephone = telephone;
```

```
        #dublu __
```

```
        self.__secretId = "3928392";
```

Varibila Secreta

```
class Student:
```

```
    def __init__(self, name, age, telephone):
```

```
        ....
```

```
        #dublu __
```

```
        self.__secretId = "3928392";
```

```
student1 = Student("Andrei", 45, 073762736276);
```

```
print(student.__secretId); -> AttributeError
```

Attribute private

Ex: `self.__secretId`

- In Python nu exista o modalitate de a face o variabila/metoda privata 100%
- Python prefixeaza numele atributului cu clasa (name mangling)
 - `__secretId` — devine —> `_Student__secretId`
- Are ca scop eliminare conflictelor de nume (doua variabile cu acelasi nume) - invatam mai tarziu
 - ex: o alta clasa `StudentPython` poate mosteni de la `Student` si crea un atribut cu acelasi nume

Varibila Secreta

```
class Student:
```

```
    def __init__(self, name, age, telephone):
```

```
        ....
```

```
        #dublu __
```

```
        self.__secretId = "3928392";
```

```
student1 = Student("Andrei", 45, 073762736276);
```

```
print(student._Student__secretId); -> functioneaza
```

Convenții de nume

- Clasa (Class):
 - numele clasei incepe cu litera mare
- Metodele (functiile), Obiectele, Atributele:
 - numele lor incep cu litera mica
- Atribute interne
 - sunt precedate in denumire de _
- Atributele private
 - sunt precedate in denumire de __

Tipul instantelor

```
class Student:
```

```
    def __init__(self, name, age):  
        self.name = name;  
        self.age = age;
```

```
class Professor:
```

```
    def __init__(self, name, age):  
        self.name = name;  
        self.age = age;
```

Tipul instantelor

```
class Student:
    def __init__(self, name, age):
        self.name = name;
        self.age = age;
```

```
class Professor:
    def __init__(self, name, age):
        self.name = name;
        self.age = age;
```

```
std = Student("Andrei", 45);
prf = Professor("Andrei", 45);
```


Tipul instantelor

```
class Student:
```

```
    def __init__(self, name, age):  
        self.name = name;  
        self.age = age;
```

```
class Professor:
```

```
    def __init__(self, name, age):  
        self.name = name;  
        self.age = age;
```

```
std1 = Student("Andrei", 45);
```

```
std2 = Student("George", 25);
```

```
prf1 = Professor("Andrei", 45);
```

```
prf2 = Professor("George", 25);
```

Tipul instantelor

```
std1 = Student("Andrei", 45);  
std2 = Student("George", 25);  
prf1 = Professor("Andrei", 45);  
prf2 = Professor("George", 25);  
print(type(std1) == type(std2));  
print(type(std1) == type(prf1));  
print(type(prf2) == type(prf1));
```

Tipul instantelor

```
std1 = Student("Andrei", 45);  
std2 = Student("George", 25);  
prf1 = Professor("Andrei", 45);  
prf2 = Professor("George", 25);  
print(type(std1) == type(std2)); #True  
print(type(std1) == type(prf1)); #False  
print(type(prf2) == type(prf1)); #True
```

Tipul instantelor

```
std1 = Student("Andrei", 45);  
std2 = Student("George", 25);  
prf1 = Professor("Andrei", 45);  
prf2 = Professor("George", 25);  
print(isinstance(std1, Student));  
print(isinstance(prf1, Student));  
print(isinstance(prf2, Professor));
```

Tipul instantelor

```
std1 = Student("Andrei", 45);  
std2 = Student("George", 25);  
prf1 = Professor("Andrei", 45);  
prf2 = Professor("George", 25);  
print(isinstance(std1, Student)); #True  
print(isinstance(prf1, Student)); #False  
print(isinstance(prf2, Professor)); #True
```

Tipul instantelor

- type
 - returneaza clasa (tipul) obiectului
- isinstance
 - returneaza true sau false daca este
 - In Python orice este un obiect
 - `isinstance(variabila, object)` -> mereu true

Tipul instantelor

```
std1 = Student("Andrei", 45);  
std2 = Student("George", 25);  
prf1 = Professor("Andrei", 45);  
prf2 = Professor("George", 25);  
print(isinstance(std1, Student)); #True  
print(isinstance(prf1, Student)); #False  
print(isinstance(prf2, Professor)); #True  
print(isinstance(prf2, object)); #True
```

Variabile statice

- Metode si variable statice apartin doar de clasa si nu de un obiect
 - Nu au nevoie de crearea unui obiect pentru a exista, ele exista la nivel de clasa

Variabile statice

```
class Student:
```

```
    STUDENT_TYPE = ["Python", "Java", "JS",  
                    "WebDesign"];
```

```
    def __init__(self, name, age):
```

```
        self.name = name;
```

```
        self.age = age;
```

Variabile statice

```
class Student:
```

```
    STUDENT_TYPE = ["Python", "Java", "JS", "WebDesign"];
```

```
    def __init__(self, name, age, type):
```

```
        self.name = name;
```

```
        self.age = age;
```

```
        if (not type in Student.STUDENT_TYPE):
```

```
            raise ValueError(f"{type} is not valid type")
```

```
        else:
```

```
            self.type = type;
```

Metode ale clasei (class methods)

- Metodele au intalnite la toate instantele de obiecte

Metode ale clasei (class methods)

```
class Student:
```

```
    STUDENT_TYPE = ["Python", "Java", "JS",  
    "WebDesign"];
```

```
    @classmethod
```



Decorator

```
    def getstudenttype(cls):
```

```
        return cls. STUDENT_TYPE
```

Metode ale clasei (class methods)

```
class Student:
```


```
    STUDENT_TYPE = ["Python", "Java", "JS",  
    "WebDesign"];
```

```
@classmethod
```

```
def getstudenttype(cls):
```

```
    return cls. STUDENT_TYPE
```

Clasa (cls), nu
obiectul (self)



EXERCITIU-(STATIC)

1. Creati o clasa Depozit pentru a depozita mancarea.
2. Depozitul va contine tipuri de mancare si capacitatea maxima.