Tema 10.
Clase și obiecte.
Atributele obiectelor

Ce ne așteaptă?

1. Paradigma programării orientate pe obiecte
2. Clase și obiecte
3. Constructorul claselor
4. Variabile în clase Python

1. Ce reprezintă programarea obiectată pe obiecte?
2. Care sunt termenii asociați POO?
3. Cum se definește o clasă?
4. Cum se creează un obiect?
5. Ce reprezintă constructorul unei clase?
6. Ce reprezintă parametrizarea constructorului?
7. Ce reprezintă variabilele de instanță?
8. Cum se declară variabilele de instanță?
9. Cum se accesează variabilele de instanță?
10. Ce reprezintă variabilele statice?
11. Cum se declară variabilele statice?
12. Cum se accesează variabilele de instanță?
13. Ce reprezintă variabilele locale ale unei clase?

1. Paradigma programării orientate pe obiecte

Noțiuni generale

- **Paradigmă de programarea – stil de scriere a unui program**

- **Programarea orientată pe funcții (POF) – programul este construi în jurul funcțiilor**

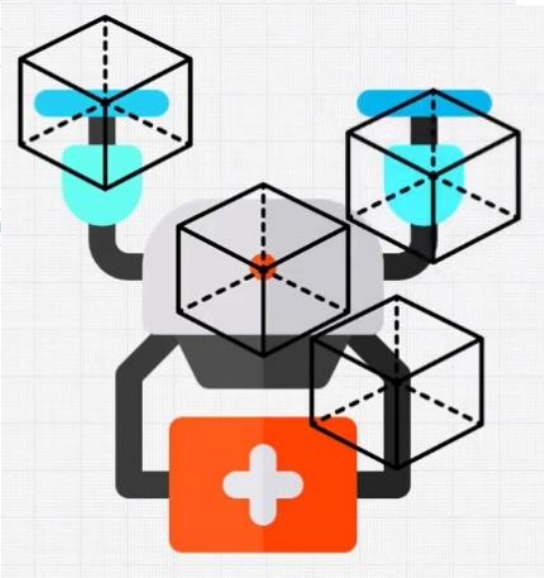
  - Limbajele POF: Erlang, Haskell și Sakaila

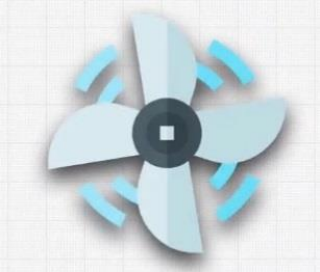
- **Programarea orientată pe proceduri (POP) – programul se divizează în sarcini pentru care se realizează diverse proceduri, funcții și rutine**

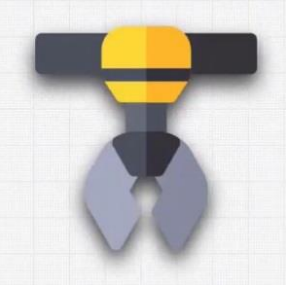
  - Limbajele POP: C, Pascal Fortran, etc

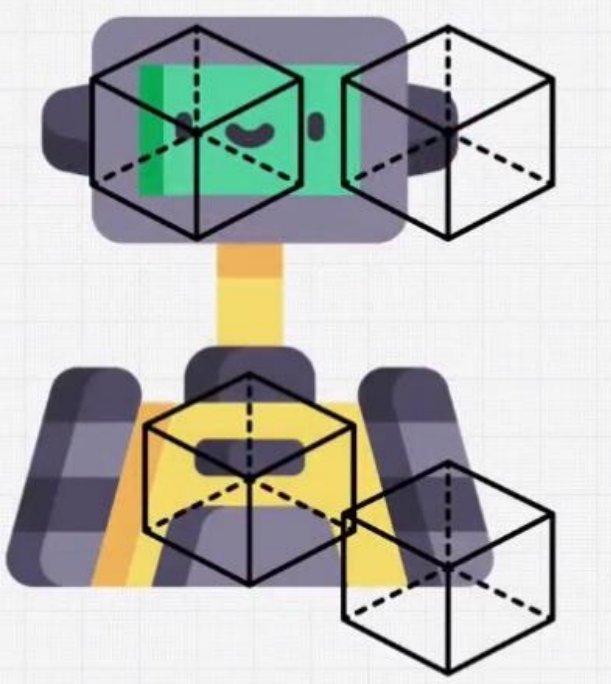
- **Programarea orientată pe obiecte (POO) – programul are la bază conceptul de clasă și obiecte**

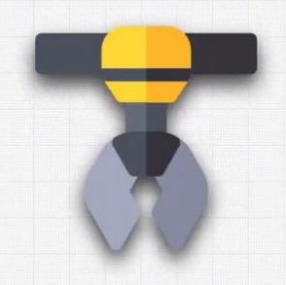
  - Limbajele POP: C++, Java, Python, Java Script, etc











Termeni asociați POO

- **POO ale la bază următorii temeni:**

- clasă – șablon în care se specifică atributele și comportamentele obiectelor

- obiect – o instanță a unei clase

- încapsulare – ascunderea logicii interioare

- abstractizare – generalizarea atributelor și comportamentelor specifice

- moștenire – obținerea atributelor și comportamentelor dintr-o clasă mai generală

- polimorfism – adaptarea comportamentelor generale propriilor necesități

2. Clase și obiecte

Definirea claselor

- **Sintaxa definirii claselor**

```
class NumeClasă:
    '''Documentatie decriere'''
    Constructori
    Variabile
        1. de instanță
        2. statice
        3. locale
    Metode
        1. de instanță
        2. statice
        3. de clasa
```

- **Exemplu de clasă**

```
class Exemplu:
    def display(self):
        print(" Un exemplu de obiect al clasei Exemplu")
```

Obiecte în Python

- **Sintaxa creării obiectelor**

  nume_obiect = NumeClasă()

- **Sintaxa apelării variabilelor și metodelor cu ajutorul obiectului:**

  nume_obiect.variabilă
  nume_obiect.metodă()

- **Exemplu de creare a obiectelor:**

```
class Om:
    def gandeste(self):
        print("Un exemplu de obiect al clasei Om")

persoana_1=Om()
persoana _1.gandeste()

persoana_2=Om()
persoana_2.gandeste()
```

3. Constructorul claselor

Noțiune de constructor

- Constructorul - o metodă specială a clasei ce se apelează implicit la crearea obiectelor
- Constructorul se apelează după fixarea variabilele statice ale clasei
- Sintaxa :

def __init__(self, *parametri*):
corpul constructorului

- Constructorul nu este obligatoriu, Python creează automat unul fără parametri
- Exemplu de apelarea implicită a constructorului fără parametri

```
class Om:
    def __init__(self):
        print("Constructorul s-a apelat implicit la crearea obiectului")

persoana = Om()
```

Parametrizarea constructorului

- Exemplu de constructor fără parametri și fără variabile

```
class Om:
    def __init__(self):
        print(„Constructor simplu")
persoana =Om()
```

- Exemplu de constructor fără parametri dar cu variabile

```
class Om:
    def __init__(self):
        Om.num_maini = 2
        print(f'Un obiect al clasei Om are {Om.num_maini} maini')
persoana_1=Om()
persoana_2=Om()
```

- Exemplu de constructor cu parametri și cu variabile

```
class Om:
    def __init__(self, nume):
        Om.num_maini = 2
        self.nume=nume
        print(f'{self.nume} are {Om.num_maini} maini')
persoana_1=Om("Ion")
persoana_2=Om("Vasile")
```

4. Variabile în clase Python

Tipuri de variabile în clase

- Variabile din clase definesc atributele obiectelor acestei clase

- Variabile de instanță – la nivelul obiectului
- Variabile statice – la nivelul clasei
- Variabile locale – la nivelul metodei

Declararea variabilelor de instanță în constructor

- Variabile de instanță - variabile a căror valoare diferă de la obiect al obiect
- Exemplu de declararea a variabilelor de instanță în constructor

```
class Om:
    def __init__(self, nume, inaltime):
        self.nume=nume
        self.inaltime=inaltime

persoana = Om("Nicolae", 164)
print(persoana.nume)
print(persoana.inaltime)
```

Declararea variabilelor de instanță utilizând o metodă de instanțiere

- **Exemplu de declararea a variabilelor de instanță utilizând o metodă de instanțiere**

```
class Om:
    def met(self, nume, inaltime):
        self.nume=nume
        self.inaltime=inaltime

persoana =Om()
persoana.met("Alexandru", 177)
print(persoana.nume)
print(persoana.inaltime)
```

Declararea variabilelor de instanță utilizând numele obiectului

- **Exemplu de declararea a variabilelor de instanță utilizând numele obiectului**

```
class Om:
    def gandeste(self):
        print("O metoda a clasei Om")

persoana = Om()
persoana.nume = "Sergiu"
persoana.inaltime = 180
print(persoana.nume)
print(persoana.inaltime)
persoana.gandeste()
```

Accesarea variabilei de instanță cu ajutorul *self*

- **Variabila de instanță se accesează cu ajutorul *self* doar în interiorul clasei**

- **Exemplu de utilizare**

```
class Om:
    def __init__(self, nume, inaltime):
        self.nume=nume
        self.inaltime=inaltime
    def afisare(self):
        print(self.nume)
        print(self.inaltime)

persoana =Om("Mihai", 192)
persoana.afisare()
```

Accesarea variabilei de instanță cu ajutorul numelui obiectului

- Variabila de instanță se accesează cu ajutorul numelui obiectului în afara clasei
- Exemplu de utilizare

```
class Om:
    def __init__(self, nume, inaltime):
        self.nume=nume
        self.inaltime=inaltime

persoana =Om("Andrei", 173)
print(persoana.nume)
print(persoana.inaltime)
```

Declararea variabilelor statice în constructor

- Variabile statice - variabile a căror valoare este constantă pentru toate obiecte clasei
- Variabile statice se pot accesa atât cu numele clasei cât și numele obiectului. Se recomandă numele clasei
- Exemplu de declararea a variabilelor statice în constructor

```
class Om:
    def __init__(self, nume):
        self.nume=nume
        Om.num_maini=2

persoana = Om(„Gheorghe")
print(Om.num_maini)
print(persoana.num_maini)
```

Declararea variabilelor statice în interiorul clasei și în metoda de instanța

- Exemplu de declararea a variabilelor statice direct în clasă

```
class Om:
    num_maini=2
    def gandeste(self):
        print("O metoda a clasei Om")

print(Om.num_maini)
```

- Exemplu de declararea a variabilelor statice utilizând o metodă de instanțiere

```
class Om:
    def met(self):
        Om.num_maini=2

persoana = Om()
persoana.met()
print(Om.num_maini)
```

Declararea variabilelor statice cu metoda de clasă

- **Exemplu de declararea a variabilelor statice cu metoda de clasă utilizând variabila *cls***

```
class Om:
    @classmethod
    def met(cls):
        cls.num_maini=2

persoana = Om()
persoana.met()
print(Om.num_maini)
```

- **Exemplu de declararea a variabilelor statice cu metoda de clasă utilizând numele clasei**

```
class Om:
    @classmethod
    def met(cls):
        Om.num_maini=2

persoana = Om()
persoana.met()
print(Om.num_maini)
```

Declararea variabilelor statice cu o metoda statică

- Exemplu de declararea a variabilelor statice cu o metoda statică

```
class Om:
    @staticmethod
    def met():
        Om.num_maini=2

Om.met()
print(Om.num_maini)
```

Variabile locale

- Variabilele locale sunt declare și accesate doar în interiorul unei metode
- Variabilele locale se declară fără variabila self sau numele clasei

```
class Om:
    def merge(self):
        incaltaminte = 42
        print(f"Persoana cind merge poarta incaltaminte {incaltaminte}")
    def cumpara(self):
        print(f"Persoana cumpara incaltaminte {incaltaminte}")

persoana = Om()
persoana.merge()
persoana.cumpara()
```